

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Grado

**DESARROLLO DE UNA PRUEBA DE
CONCEPTO DE UN DETECTOR DE
INTRUSIONES PARA SU APLICACIÓN EN
PRÁCTICAS DE LABORATORIO**

**(Development of a proof of concept of an
intrusion detector for its application in
laboratory practices)**

Para acceder al Título de

***Graduado en
Ingeniería de Tecnologías de Telecomunicación***

Autor: Naiara Rasilla Villegas

Septiembre – 2020

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Naiara Rasilla Villegas

Director del TFG: Alberto Eloy García Gutiérrez

Título: “Desarrollo de una prueba de concepto de un detector de intrusiones para su aplicación en prácticas de laboratorio”

Title: “Development of a proof of concept of an intrusion detector for its application in laboratory practices”

Presentado a examen el día: 18/09/2020

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente: Irastorza Teja, José Angel

Secretario: García Gutiérrez, Alberto Eloy

Vocal: Medina Pasaje, Julio Luis

Este Tribunal ha resuelto otorgar la calificación de:

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado N°
(a asignar por Secretaría)

Resumen

El creciente desarrollo de las TIC ha derivado en prácticas que atentan contra la seguridad de los usuarios y los sistemas. Esto frecuentemente se manifiesta en forma de intrusiones no deseadas que pueden ser identificadas con los Sistemas de Detección de Intrusiones. Además, con el fin de facilitar la detección y permitir que su análisis esté al alcance de casi cualquiera, existen una serie de herramientas que proporcionan una interfaz gráfica y recopilan los datos y los clasifican según el interés del usuario.

El trabajo se centra en la selección e implementación del Sistema de Detección de Intrusiones más adecuado para responder ante ciertos ataques que son simulados, así como en la búsqueda de herramientas adicionales que, en combinación con ese sistema, faciliten la comprensión de los datos por usuarios no avanzados y que permitan una implementación en prácticas de laboratorio.

Palabras clave

Snort, ciberseguridad, detección, intrusiones, vulnerabilidades.

Abstract

The increasing development of the ITCs has lead to practices that threaten the security of the users and the systems. This usually shows up in form of unwanted intrusions which can be identified with Intrusion Detection Systems. Moreover, with the aim of making the detections easier and making the analysis suitable almost for everyone, there are some additional tools that provide a graphic interface and collect data and classify it according to the user needs.

This project focuses on the selection and implementation of the most suitable Intrusion Detection System to respond to certain attacks that are previously simulated, as well as on the searching of additional tools which, together with this system, make the interpretation of the information by starter users easier and allow the implementation of all in laboratory practices.

Key words

Snort, cybersecurity, detection, intrusions, vulnerabilities.

Índice de Contenidos

1. Introducción	1
1.1. Motivación y objetivos	1
1.2. Organización del documento	2
2. Aspectos teóricos	5
2.1. Qué son los IDS.....	5
2.2. Un IDS ideal.....	5
2.3. Clasificación de los IDS	6
2.3.1. Según qué sistemas vigilan	6
2.3.1.1. HIDS: Host-Based	6
2.3.1.2. NIDS: Network.....	7
2.3.2. Según cómo actúan	9
2.3.2.1. Detección de anomalías	9
2.3.2.2. Detección de usos indebidos	10
2.3.3. Según el tipo de respuesta.....	11
2.4. Elección de IDS.....	11
2.4.1. Snort.....	11
2.4.2. Suricata.....	12
2.4.3. Discusión.....	13
3. Aspectos prácticos	15
3.1. Snort.....	15
3.1.1. Entorno e instalación	15
3.1.2. Primeros pasos con Snort.....	16
3.1.2.1. Sniffer sobre captura o packet logger	16
3.1.2.2. Sniffer en tiempo real.....	17
3.1.2.3. Snort como NIDS	18
3.2. Herramientas en torno a Snort	19
3.2.1. IDS Policy Manager	20
3.2.2. Cyberprobe	23
3.2.3. Snorby	26

3.2.4. BASE y PMGraph	30
4. Ejemplo de aplicación	35
4.1. Plataforma desplegada.....	35
4.2. Pruebas de intrusión	35
4.2.1. Ataque por escaneo de puertos.....	35
4.2.2. Ataque por fuerza bruta	37
4.2.3. Ataque DoS	39
5. Conclusiones y líneas futuras.....	43
Bibliografía	45

Índice de Figuras

Figura 1: Cabecera IPv4	7
Figura 2: Estructura de un segmento TCP	8
Figura 3: Banner de Snort que muestra la versión instalada.....	16
Figura 4: Resumen de paquetes detectados en escaneo de captura.....	17
Figura 5: ECHO y ECHO REPLY capturados en tiempo real	18
Figura 6: Resumen paquetes detectados en escaneo a tiempo real	18
Figura 7: Validación de archivo de configuración	19
Figura 8: Alertas ante paquetes ICMP detectados.....	19
Figura 9: Oink Code en IDS Policy Manager	21
Figura 10: Creación de nueva política en IDS Policy Manager.....	21
Figura 11: Creación de nueva alerta en IDS Policy Manager.....	22
Figura 12: Creación de sensor en IDS Policy Manager.....	22
Figura 13: Alerta creada en IDS Policy Manager	23
Figura 14: Sensor creado en IDS Policy Manager	23
Figura 15: Archivo c.cfg de Cyberprobe.....	24
Figura 16: Mensajes de verificación de Cyberprobe	25
Figura 17: Alerta mostrada por Cyberprobe	26
Figura 18: Configuración de parámetros Snorby	26
Figura 19: Dirección IP de la aplicación web	27
Figura 20: Entrada a Snorby	27
Figura 21: Dashboard Snorby.....	28
Figura 22: Eventos en Snorby	29
Figura 23: Diagrama origen detecciones Snorby	30
Figura 24: Mensaje bienvenida EasyIDS	31

Figura 25: Cambio de clave en EasyIDS	31
Figura 26: Servicios activos en EasyIDS	31
Figura 27: Protocolos detectados en BASE	32
Figura 28: Alertas detectadas en BASE	32
Figura 29: Detalles de alerta en BASE	32
Figura 30: Gráfico alertas en BASE	33
Figura 31: Tráfico en eth0 con PMGraph	34
Figura 32: Banner bienvenida a Metasploit	36
Figura 33: Port scanning sin Snort	36
Figura 34: Alerta escaneo de puertos Snort	37
Figura 35: Registro de escaneo de puertos en Snorby	37
Figura 36: Credenciales descubiertas por xHydra	38
Figura 37: Alerta ataque por fuerza bruta en Snort	39
Figura 38: Gráfico alertas ataque fuerza bruta en BASE	39
Figura 39: Esquema ataque DoS simple	40
Figura 40: Intercambio de paquetes en ataque DoS	41
Figura 41: Esquema ataque DoS múltiples IPs	41
Figura 42: Intercambio de paquetes en ataque DoS aleatorizado	41
Figura 43: Alerta DoS en Snort	42
Figura 44: Gráfico alertas ataque DoS en BASE	42

Índice de Tablas

Tabla 1: Comparación entre Snort y Suricata	13
---	----

Acrónimos

ACK: Acknowledgement

BASE: Basic Analysis and Security Engine

CGI: Common Gateway Interface

CPU: Central Processing Unit

DDoS: Distributed Denial of Service

DF: Don't Fragment

DMZ: Demilitarized Zone

DNS: Domain Name Service

DoS: Denial Of Service

GPL: General Public License

HIDS: Host Intrusion Detection System

HTTP: Hypertext Transfer Protocol

ICMP: Internet Control Message Protocol

ICT: Information and Communication Technologies

ID: Identifier

IDS: Intrusion Detection System

IP: Internet Protocol

IPS: Intrusion Prevention System

IPv4: Internet Protocol version 4

JSON: JavaScript Object Notation

MF: More Fragments

MTU: Maximum Transmission Unit

NIDS: Network Intrusion Detection System

OISF: Open Information Security Foundation

OS: Operating System

PDF: Portable Document Format

SID: Signature ID

SMB: Server Message Block

SO: Shared Object

SSH: Secure Shell

SYN: Synchronice

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

VM: Virtual Machine

VRT: Vulnerability Research Team

YAML: YAML Ani't Markup Languag

1. Introducción

En el mundo informático actual hay una lucha permanente por preservar la seguridad de los usuarios que hacen uso de las tecnologías. Estén o no conectadas a la red, es importante tener conciencia de que en cualquier entorno puede encontrarse una vulnerabilidad que comprometa la integridad o privacidad de un recurso. Estas brechas de seguridad pueden ser utilizadas para sacar provecho de ellas desde por un pirata informático o hacker hasta por un usuario del propio sistema con conocimientos de seguridad que quiere acceder a privilegios que no le corresponden.

Es en estas circunstancias cuando toma valor el concepto de seguridad informática o ciberseguridad, que es el área de la informática que se enfoca en la protección de toda la estructura computacional, desde una computadora hasta un servidor, un sistema eléctrico, una red, un dispositivo móvil o los propios datos. Además, es ante este tipo de situaciones cuando entran en juego los sistemas de detección de intrusiones, que hacen los rastreos y análisis necesarios para detectar esos intentos de acceso no deseados y son lo suficientemente autónomos como para aprender de los intrusos ya conocidos y facilitarse las futuras detecciones.

En combinación con los sistemas de detección de intrusiones existentes pueden emplearse otros softwares o herramientas que nos permitan tomar ventaja del conocimiento del ataque que puede aportar un IDS y así poder prevenirlo o subsanar los daños que haya podido ocasionar el intruso. Se trata de una práctica con decenas de años de recorrido, pero a día de hoy aún se siguen inventando y mejorando herramientas que pretenden procesar cada vez más la información que se obtenga del IDS y mostrarla de la manera más gráfica posible para que cualquiera pueda interpretarla y que bajo ningún concepto el manejo de estos sistemas suponga un obstáculo para la seguridad.

1.1. Motivación y objetivos

Nuestra sociedad es cada vez más dependiente de las tecnologías, lo que ha supuesto una proliferación del empleo de ordenadores y de las redes de comunicaciones, y con ello se ha acrecentado la picaresca de aquellos que se quieren beneficiar de fallos o vulnerabilidades que encuentren en los sistemas, tanto físicos como virtuales, para acceder a datos privados, modificar archivos o configuraciones en remoto, suplantar identidades, etc.

Estas personas son conocidas como piratas informáticos o hackers y, con la necesidad de prevenir sus actuaciones, han surgido los Sistemas de Detección de Intrusiones.

El objetivo de este trabajo recae en esta creciente necesidad, especialmente por parte de las empresas, de proteger su infraestructura, lo que dirige de manera intrínseca al análisis de los sistemas de detección de intrusiones existentes hoy en día. Para ello es necesario realizar un estudio teórico sobre cómo funcionan dichos sistemas en los diferentes escenarios que se puede encontrar y decidir cuál es el óptimo para cubrir las necesidades de este proyecto. Una vez centrado el objetivo, procede simular éstos escenarios junto con intrusiones reales para probar de manera práctica su funcionamiento y con el ánimo de descubrir cómo alerta al usuario ante la detección de estas amenazas informáticas.

Además, se pretenden implementar una serie de herramientas gráficas que aporten datos de una manera más visual e interactiva, para que su uso no se limite únicamente a usuarios avanzados y que la protección frente a amenazas en ningún momento suponga un obstáculo.

Dando un paso más allá, todo este proyecto podrá ser aplicado con fines educativos para formar a futuros estudiantes en materias de ciberseguridad y familiarizarlos con el uso de los sistemas detectores de intrusiones. Su objetivo es realizar pruebas de ataque en equipos del laboratorio, con el fin de no comprometer la seguridad del resto de equipos, para su futura implementación en prácticas en la universidad.

1.2. Organización del documento

La organización de este documento, tras este capítulo de introducción inicial, se estructura en otros cuatro capítulos que tratan el contenido de la siguiente manera:

A continuación, antes de desarrollar el proceso de implementación, se presentan en el segundo capítulo los aspectos teóricos comunes a todos los Sistemas de Detección de Intrusiones, se les clasifica y, tras identificar dos sistemas que cumplen los requisitos, se realiza una elección del más apto para llevar a cabo este proyecto.

El tercer capítulo engloba todos los aspectos prácticos de la instalación y pruebas de Snort en sus diferentes modalidades de trabajo, así como la instalación de algunas herramientas que proporcionan un valor añadido a Snort.

En el cuarto capítulo, se realizan tres tipos de intrusiones reales a otras máquinas y se estudia cómo Snort las puede detectar y cómo se visualizan con alguna de esas herramientas adicionales con las que se trabaja en la sección anterior.

Por último, se realiza en el quinto capítulo una conclusión en base a los resultados observados por parte de Snort y, en general, a la información recopilada anteriormente. Además se proponen posibles líneas de investigación para futuros proyectos.

2. Aspectos teóricos

A continuación, se hace una breve introducción de los principales términos y principios teóricos relacionados con la seguridad informática, especialmente los relacionados con la posibilidad de detectar agresiones incluso antes de que éstas ocurran.

2.1. Qué son los IDS

Un Sistema de Detección de Intrusiones, o IDS, es una herramienta de seguridad que trata de detectar o monitorizar intentos de comprometer la seguridad de un sistema informático. Para llevar a cabo esta tarea, su defensa se basa en algo más que el clásico control de acceso, ya que analizan el tráfico de la red y examinan paquetes, buscando patrones previamente definidos que puedan permitir la detección de ataques en fases tempranas. Los IDS no están diseñados para enfrentarse directamente a los intrusos, que pueden ser desde un pirata informático que pretende comprometer la integridad, confidencialidad o disponibilidad de un recurso, hasta un usuario no autorizado que intenta obtener privilegios que no le corresponden.[1]

2.2. Un IDS ideal

Un buen IDS debe tener la capacidad de ejecutarse continuamente sin necesidad de tener supervisión humana, es decir, deben estar configurados de manera que su trabajo habitual sea transparente ante los operadores del entorno informático.

También es importante que el sistema sea aceptable para las personas que trabajen habitualmente en el entorno. Por ejemplo, si supone una excesiva sobrecarga para el sistema, si tiene un porcentaje alto de falsos positivos, o tiene excesivos logs (que registran las actividades detectadas), los operadores podrían optar por ignorarlo y se podrían pasar por alto situaciones que comprometan la seguridad.

Por otro lado, es deseable una buena adaptabilidad, o lo que es lo mismo, la capacidad para adaptarse rápidamente a cambios en el entorno de trabajo, ya que los sistemas informáticos no son estáticos. Por último, siempre será deseable una alta tolerancia a fallos, así como una capacidad óptima de respuesta ante situaciones inesperadas.[1]

2.3. Clasificación de los IDS

Los IDS son capaces de detectar una intrusión en diversas situaciones y de diferente modo. Es por eso por lo que se les clasifica de tres formas distintas, aunque las principales sean las que los distinguen según qué sistemas vigilan y por cómo lo hacen.

2.3.1. Según qué sistemas vigilan

Esta primera clasificación, se subdivide en dos niveles de protección diferentes, aunque con un objetivo común: alertar de actividades sospechosas y, en algunos casos, proporcionar una respuesta automática a las mismas. Hoy en día, aunque una de estas categorías engloba a los sistemas más utilizados, ambas pueden ser igualmente necesarias para crear un esquema de detección con la máxima efectividad.

2.3.1.1. HIDS: Host-Based

Estos fueron los primeros IDS en aparecer en el mundo de la seguridad informática. Protege únicamente a la máquina sobre la que se ejecuta, por lo que haría falta un HIDS para monitorizar cada sistema individual. Para llevar a cabo su tarea se emplean analizadores de *logs* (que registran las actividades que se llevan a cabo en el sistema), generados por diferentes aplicaciones o por el propio *kernel* del sistema operativo, prestando atención a los registros sospechosos.

También se utilizan los verificadores de integridad de ficheros importantes, empleando funciones resumen o *hash* de los ficheros a monitorizar. De esta forma, si su contenido sufre cualquier modificación, generará un resumen diferente, que al comparar con el *hash* original dará la voz de alarma. Para este tipo de verificación es importante disponer de una base de datos segura, preferiblemente separada físicamente de la máquina, en la que se albergan los datos (tamaños de ficheros, hash, etc.), que se usarán como referencia para comparar con lo que se está analizando.

Otra estrategia es el empleo de sistemas de decepción o *honeypots*, que son sistemas basados en el concepto teórico de conocimiento negativo, ya que están diseñados para recibir ataques y aplicar una estrategia de respuesta. Frecuentemente, simulan vulnerabilidades que hagan creer al atacante que ha tenido éxito y que tiene a un sistema vulnerable ante él, para, mientras tanto, monitorizar su comportamiento para aprender cuál es su forma de trabajar o recopilar información del atacante.[5]

2.3.1.2. NIDS: Network

Este tipo de sistemas monitorizan directamente los paquetes que circulan en una misma red, concretamente dentro de un mismo dominio de colisión, en busca de elementos que denoten un ataque contra alguno de los sistemas ubicados en ella. Para ello, al menos una interfaz de red de esa máquina sensor debe trabajar en modo promiscuo, analizando así todas las tramas que circulan por un segmento de la red (vayan o no destinados al equipo en cuestión) en busca de patrones sospechosos de comprometer la seguridad.[5]

Para analizar estos patrones, en ocasiones basta con fijarse en las tramas de red TCP/IP, ya que hay ciertos valores que pueden representar con gran probabilidad un ataque real. Algunos ejemplos son:

- Campos de fragmentación: La fragmentación IP permite dividir un datagrama IP en varios bloques de datos más pequeños si su tamaño sobrepasa la MTU. Para esta forma de detección basta con analizar los 3 bits del campo *Flags* de la cabecera IPv4, destacados en la Figura 1.



Figura 1: Cabecera IPv4 [2]

De menos a más significativo: el primer bit siempre va a '0'; el segundo bit (DF) indica con un '1' que se puede fragmentar el datagrama, y con un '0' si no; por último, el tercer bit (MF) si se encuentra a '1' indica que hay más fragmentos en camino, por lo que todos los fragmentos llevarán un '1', excepto el último, que irá a '0'.

En ocasiones, los atacantes utilizan combinaciones de estos bits, por lo que puede ser un indicativo útil que haga sospechar que alguien trata de comprometer la seguridad de nuestro sistema. Por ejemplo, si siempre se reciben datagramas con el bit MF a '1' el sistema podría

quedarse esperando a un último paquete que no va a llegar. O si recibe uno con DF a '0' que en teoría no puede fragmentarse, pero lleva MF a '1' como si no fuera el último fragmento que va a recibir.[2]

- Dirección de origen y destino: Es motivo de sospecha si el tráfico con origen en nuestra DMZ tiene como destino la red que se intenta proteger, ya que, normalmente, el tráfico proveniente de la DMZ solo se permite hacia la red externa o Internet. También lo es cuando una petición llega desde Internet a una máquina concreta de nuestra red interna que no ofrece servicios directos al exterior, por ejemplo, un servidor que alberga una base de datos privada.
- Puertos de origen y destino: Los puertos de origen y destino permiten, por un lado, detectar la presencia de troyanos, y por otro, reaccionar ante los barridos de puertos o la presencia de servidores no autorizados en nuestra red.
- Flags TCP: Los segmentos TCP tienen varios campos de booleanos, como son SYN y FIN, representados en la Figura 2. El bit SYN se emplea para establecer una nueva conexión TCP, y FIN sirve para finalizarla.[3]



Figura 2: Estructura de un segmento TCP

Una vez más, hay ciertas combinaciones que pueden ser indicios de un ataque, como son los dos bits activados simultáneamente. Esto representaría a una conexión que trata de abrirse y cerrarse al mismo tiempo.

- Campo de datos: Los atacantes pueden aprovecharse de que los cortafuegos normalmente se fijan en las tramas cuya cabecera sea sospechosa, pero no en los datos que lleva. Así podrían intentar una intrusión en nuestro sistema a través de una inocente petición de tipo GET.

Estos son algunos ejemplos de intrusiones que podrían detectarse analizando los campos de una trama TCP/IP. Sin embargo, no siempre es suficiente, ni necesario, analizar tramas o patrones, ya que hay ataques que se podrían descubrir por la presencia de repeticiones, como por ejemplo el barrido de puertos horizontales o verticales, o las negaciones de servicio distribuidas DDoS, o por anomalías, como puede ser el tráfico excesivo entre dos sistemas (atacante y atacado).

Por último, y en analogía con los *honeypots* en los HIDS, procede hablar de las *honeynets*. Al igual que los anteriores, se tratan de sistemas diseñados para ser comprometidos y están formados por sistemas reales que, al ser atacados, monitorizan la actividad del atacante. Su principal diferencia es que son redes completas con sistemas reales, es decir, no simulan ninguna vulnerabilidad, sino que trabajan tal y como lo haría un sistema en una estación de trabajo normal. Como el objetivo de las *honeynets* no es la detección, sino conocer más sobre el atacante en un entorno semireal y poder emplear lo aprendido para mejorar sus sistemas reales, es importante que la información que se recoja nunca se almacene dentro del perímetro de la *honeynet*, ya que muy probablemente el atacante podría descubrirlo, y como consecuencia, lo eliminaría o modificaría con el fin de autoprotegerse. Además, debe garantizarse su aislamiento y la capacidad de control sobre esta red, y que el atacante no pueda tomar ventaja de ella para saltar hacia otra red y atacar a otras máquinas.

2.3.2. Según cómo actúan

La segunda forma de clasificación de los IDS lo hace en función de cómo actúan: basándose en la detección de anomalías o en la detección de usos indebidos.

2.3.2.1. Detección de anomalías

Identifican como una intrusión a cualquier desviación excesiva del comportamiento habitual de un sistema, basándose en los conceptos de conocimiento negativo y conocimiento positivo. El propio sistema puede analizar los datos de los que dispone para obtener elementos estadísticos, como medias o varianzas, con las que define un perfil de comportamiento de los usuarios, que va actualizándose y comparándose con el último perfil registrado, en busca de desviaciones.

Otra modalidad de detección de anomalías es la basada en especificaciones, que consiste en proporcionar reglas al sistema que definan perfiles de comportamiento normales o deseados, las cuales deben ser lo

suficientemente precisas como para delatar los comportamientos realmente intrusivos, sin demasiadas falsas alarmas, ni siendo demasiado permisivas.[1]

2.3.2.2. Detección de usos indebidos

Consiste en especificar las posibles intrusiones que pueden poner en riesgo la seguridad de un sistema y esperar a que ocurran.

Se pueden emplear los llamados sistemas expertos, en los que se describen las intrusiones como reglas del tipo *if - then* (condición – acción), basándose en el análisis de los registros de cualquier sistema Unix. Por ejemplo, si un usuario con una sesión abierta en una máquina (y se encuentra un log en el sistema que lo registra) intenta acceder desde otra máquina, se realizaría la acción definida en la regla, normalmente alertar al responsable de seguridad.

Otra modalidad es la basada en el análisis de transición entre estados; en la que se entiende una intrusión como una secuencia de eventos que llevan al atacante desde un estado previo a la intrusión, a uno en el que ya se ha producido el ataque. Así, si se logra captar ,como una imagen, los parámetros del sistema en esa transición de eventos, será posible detener la intrusión antes de que se haya llevado a cabo por completo. Uno de los sistemas de detección basados en el análisis de transición entre estados es *Ustat*, que maneja una base de datos que recoge las descripciones de estados y de transiciones sospechosas entre ellos, y emplea un motor de decisiones para actuar ante una intrusión.

Una alternativa más ventajosa es la detección basada en el uso de reglas de comparación y emparejamiento de patrones, en la que se entiende que se está comprometiendo la seguridad del sistema al recibir un patrón predefinido de una intrusión, sin importar el estado en el que se encuentre. El sistema más conocido basado en el *pattern matching* es *Snort*, que se encarga de analizar el tráfico y registrar una alarma en el sistema si detecta en él algún patrón de los definidos.

Por último, existen los sistemas de detección de intrusos basados en modelos, de funcionamiento similar a la basada en el análisis de transición entre estados. Éstos emplean una base de datos de escenarios que definen los ataques, en lugar de una con transiciones de estados, y necesitan varios procesos para analizar registros, actualizar escenarios, etc.[1]

2.3.3. Según el tipo de respuesta

Aunque este tipo de clasificación es minoritaria en comparación con las otras dos, los IDS también pueden dividirse en activos y pasivos. Los activos son aquellos que trabajan continuamente y en tiempo real en busca de posibles ataques. En cambio, los pasivos analizan las vulnerabilidades de forma periódica, ya que un administrador debe ejecutarle regularmente, ya sea de manera manual o automática, para comprobar que sus sistemas no presentan problemas de seguridad.[5]

2.4. Elección de IDS

Tras analizar los distintos tipos de IDS, para la realización de este trabajo procede centrarse en un detector de intrusiones en red, o NIDS, pues su acción no está limitada a una única máquina. Partiendo de esa condición, se discutirán las diferencias entre las dos mejores herramientas open source de detección de intrusiones en red: *Snort* y *Suricata*.

2.4.1. Snort

Es un sistema de detección de intrusiones en red (NIDS) basado en la detección de usos indebidos, de código abierto y capaz de analizar tráfico y capturar paquetes en tiempo real. Puede realizar análisis de protocolos, búsquedas de contenidos y de coincidencias, y puede usarse para detectar una variedad de ataques y sondeos, como buffer overflows (desbordamientos de búfer), escaneos furtivos de puertos, ataques a la CGI, sondeos al SMB e intentos de OS Fingerprinting (recopilación de información que permite identificar el sistema operativo), entre otros.

Se puede ejecutar en máquinas Windows y UNIX/Linux, y el hecho de que sea multiplataforma, gratuito y que sea tan versátil, lo convierte en uno de los IDS más populares, pudiéndose encontrar abundante información sobre su uso tanto de manera oficial como en foros.

Puede funcionar como sniffer, leyendo y mostrando los paquetes que atraviesan la red; como logger de paquetes, guardando en un archivo los logs para su posterior análisis; o puede hacer las tareas de un NIDS normal, con multitud de opciones de configuración.

Los elementos que componen la arquitectura de Snort, encargados de implementar estas funcionalidades son:

- Módulo de captura del tráfico: Encargado de capturar todos los paquetes de la red utilizando la librería *libpcap*, que le permite ser instalado en casi cualquier parte.
- Decodificador: Una serie de decodificadores forman las estructuras de datos con los paquetes capturados e identificar los protocolos de enlace, de red, etc.
- Preprocesadores: Pequeños programas en C que, aplicados en forma de librerías, toman la información que viaja desordenada por la red y le dan forma para que pueda ser interpretada a continuación, tras aplicar las reglas.
- Motor de detección: Analiza los paquetes y aplica las reglas definidas para detectar cualquier actividad de intrusión existente en un paquete.
- Reglas: Se emplean en el motor de detección para regir el análisis de los paquetes, y generar las alertas en caso necesario.
- Plugins de salida: Permiten definir qué, cómo y dónde se guardan las alertas y los correspondientes paquetes de red que las generaron. Pueden ser archivos de texto, bases de datos, servidor syslog, etc.
- Plugins de detección: Partes del software que son compilados con Snort y se usan para modificar el motor de detección.[7]

Con todo ello, el conjunto resulta compacto y muy versátil.

2.4.2. Suricata

Al igual que Snort, Suricata es un sistema de detección de intrusiones en red (NIDS) de código abierto muy robusto. Se puede ejecutar en máquinas Windows y UNIX/Linux y es capaz de realizar en tiempo real detección y prevención de intrusos (IDS/IPS), de monitorizar la red y de procesar offline capturas de paquetes. Para analizar el tráfico emplea reglas o firmas y posee un poderoso soporte de secuencias de comandos *Lua* para detección de amenazas complejas.

Entre sus características cabe destacar su arquitectura multi-hilo, que permite que solo con ejecutarse en una instancia el monitor balanceará su carga entre todos los procesadores disponibles que se desee. Esto le permite procesar un gran ancho de banda, de hasta 10 Gbit/s, sin afectar a su rendimiento.

Suricata también es capaz de identificar los principales protocolos de red y de controlar un gran número de formatos de archivos diferentes, así como de realizar comprobaciones MD5 para garantizar la integridad de éstos.[8][9]

2.4.3. Discusión

En la Tabla 1 se exponen las características principales de los dos IDS presentados previamente. Esto permite realizar una comparación entre ambos, que descubre que son herramientas muy similares, siendo la arquitectura multi-hilo de Suricata la diferencia más destacable.

Asimismo, Suricata tiene un mayor rendimiento ante situaciones de mayor tráfico, lo cual conlleva un uso de CPU y memoria más elevado, que en un ordenador con recursos limitados podría ser contraproducente. No obstante, la ventaja de Suricata en cuanto a rendimiento se ve reducida al no trabajar en redes congestionadas, en las que Snort también se maneja sin precisar tantos recursos computacionales para su implementación.

En referencia a Snort, cabe destacar su gran comunidad, que apoya a la continua evolución de su base de datos y facilita la búsqueda de información a través de internet, lo cual ha sido un punto clave a la hora de seleccionar IDS.[6]

En definitiva, teniendo en cuenta las necesidades del proyecto, ambos IDS resultarían válidos. Sin embargo, algunos factores relacionados con el entorno de implementación, junto con la documentación disponible, han decantado la elección a favor de Snort.

Tabla 1: Comparación entre Snort y Suricata[10]

Característica	Snort	Suricata
Desarrollador	Sourcefire (ahora Cisco)	OISF
Disponibilidad	Desde 1998	Desde 2009
Lenguaje	C	C
Plataformas	Windows, Unix/Linux	Windows, Unix/Linux
Licencia	GNU GPLv2	GNU GPL v2
Última versión estable	2.9.14.1	4.1.4
Procesamiento	Mono-hilo	Multi-hilo
IPS	Opcional	Opcional
Soporte IPv6	Sí	Sí
Reglas	VRT, Emerging Threats Rules, SO Rules	VRT, Emerging Threats Rules
Formato de output	Unified2	Unified2, JSON, YAML
Escalabilidad	Medio	Alto

Uso de CPU y memoria	Medio	Alto
Documentación	Abundante	Escasa

3. Aspectos prácticos

A continuación se hace referencia a los detalles de instalación y configuración de los sistemas utilizados durante el desarrollo de este trabajo, siendo especialmente importantes aquellos relacionados con la resolución de problemas, motivados, la mayoría de las veces, por incompatibilidades entre las diferentes versiones de software.

3.1. Snort

Tal como se ha indicado anteriormente, este software es un sistema de prevención de intrusiones de código abierto capaz de analizar el tráfico en tiempo real y registrar paquetes, con versiones para la mayoría de sistemas operativos, accesibles desde su propio dominio, <http://www.snort.org>, que centraliza todos los desarrollos de la comunidad de desarrolladores.

3.1.1. Entorno e instalación

Se dispone de un ordenador portátil Lenovo con un procesador Intel Core i7-8565U de 1,8 GHz, 8 GB de memoria RAM y un disco SSD de 512 GB, con el sistema operativo Windows 10 de 64 bits instalado.

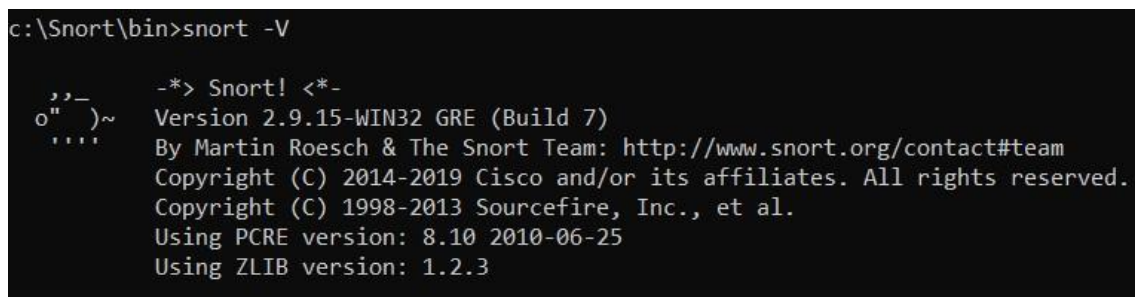
Siguiendo el propósito inicial, se trabajará con Snort sobre Windows, por lo que es necesario descargarse de la página web oficial del software los archivos para la instalación de Snort y las reglas. Además, se instalará WinPcap, que será el motor de captura de paquetes y filtrado de Snort, ya que el propio software no posee de manera nativa una utilidad para captura de paquetes.

La instalación de Snort en Windows, a diferencia de la de WinPcap, no es muy intuitiva, dado que requiere cierto grado de personalización en el entorno anteriormente descrito. En primer lugar, tras ejecutar el instalador, se deben desplazar las carpetas descargadas *rules* y *preproc-rules* al directorio raíz de Snort y, observar que en el directorio *C:\Snort\etc* hay un archivo de configuración denominado *snort.conf* que se ha creado durante su instalación.

El primer problema encontrado es el formato en el que está redactado este archivo de configuración, ya que los usuarios de Windows deben adaptarlo a su entorno, como indica el propio creador. Además, hay que realizar una serie de personalizaciones para que su funcionamiento sea tan específico como se desee, que a grandes rasgos son las siguientes:

- Se cambia la declaración de HOME_NET para designar el rango de direcciones IP que se desea proteger. Por defecto aparece como *any*, pero se introducirá la dirección IP 192. 192.1.135/24 para que proteja el equipo en cuestión.
- Se cambia la declaración de EXTERNAL_NET a !\$HOME_NET para definir como red externa a cualquier IP que difiera de la dada anteriormente.
- Se definen los path completos para ubicar todas las reglas, librerías y logs con el formato “c:\Snort\rules”, por ejemplo. [12]

Una vez editado el fichero de configuración es necesario abrir una ventana de comandos con permisos de administrador y dirigirse al directorio c:\Snort\bin para ejecutar los comandos de Snort, ya que es donde se aloja el ejecutable. Puede comprobarse de una manera rápida y sencilla que Snort ha sido correctamente instalado con el comando “*snort -V*”, que muestra la versión instalada, tal y como refleja la Figura 3.



```
c:\Snort\bin>snort -V

,,_      -*> Snort! <*-
o" )~    Version 2.9.15-WIN32 GRE (Build 7)
' ' '    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
         Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.
         Using PCRE version: 8.10 2010-06-25
         Using ZLIB version: 1.2.3
```

Figura 3: Banner de Snort que muestra la versión instalada

3.1.2. Primeros pasos con Snort

Como se ha explicado anteriormente, es conveniente el uso de un detector de intrusiones en red o NIDS para hacer el análisis más completo y configurable. Pero, en primer lugar, se han hecho una serie de escaneos más sencillos para verificar el correcto funcionamiento de Snort y familiarizarse con su uso.

3.1.2.1. Sniffer sobre captura o packet logger

Snort no solo trabaja en tiempo real, sino que puede analizar capturas previamente realizadas. Con el comando “-c” se especifica la ubicación del archivo de configuración, y con la opción “-r” le indicamos a Snort que no capture el tráfico de la tarjeta de red, sino del archivo .pcap que le indicamos.

snort -c c:\Snort\etc\snort.conf -r ping.pcap

En el ejemplo de la Figura 4 se ha capturado en Wireshark un ping a otro dispositivo de la misma red para, posteriormente, analizarlo con Snort. Se

comprueba que de los 13 paquetes capturados, 8 son del tipo ICMP. Esto puede servir de pista para detectar los típicos ataques *Smurf*, que consisten en enviar una invasión de paquetes ICMP (echo request) a una dirección broadcast desde una dirección IP origen falsa.

```

Snort processed 13 packets.
Snort ran for 0 days 0 hours 0 minutes 0 seconds
  Pkts/sec:          13
=====
Packet I/O Totals:
  Received:          13
  Analyzed:           13 (100.000%)
  Dropped:            0 (  0.000%)
  Filtered:           0 (  0.000%)
  Outstanding:       0 (  0.000%)
  Injected:           0
=====
Breakdown by protocol (includes rebuilt packets):
  Eth:                13 (100.000%)
  VLAN:               0 (  0.000%)
  IP4:                 9 ( 69.231%)
  Frag:               0 (  0.000%)
  ICMP:                8 ( 61.538%)
  UDP:                 1 (  7.692%)

```

Figura 4: Resumen de paquetes detectados en escaneo de captura

3.1.2.2. Sniffer en tiempo real

Snort puede trabajar como sniffer del tráfico en tiempo real. Empleando la opción “-v” se muestran por consola de manera instantánea los paquetes que están atravesando la red, y añadiendo “-i [interfaz]” se indica sobre cuál se desea capturar. Para consultar la lista de interfaces disponibles puede usarse previamente la opción “-W”.

```
snort -v -i 2
```

En la Figura 5 se muestran dos de los paquetes capturados en este modo en los que se indica la dirección origen y destino. Finalmente, Snort siempre muestra un resumen como el de la Figura 6 con toda la información relativa a los paquetes y protocolos observados. Este modo tiene un principal inconveniente, y es que como el propio manual de Snort indica, puede llegar a ser muy lento y puede haber una pérdida de paquetes.

```

01/26-17:44:44.276038 192.168.1.135 -> 192.168.1.130
ICMP TTL:128 TOS:0x0 ID:35578 IpLen:20 DgmLen:60
Type:8 Code:0 ID:1 Seq:965 ECHO
=====
01/26-17:44:44.282309 192.168.1.130 -> 192.168.1.135
ICMP TTL:64 TOS:0x0 ID:10215 IpLen:20 DgmLen:60
Type:0 Code:0 ID:1 Seq:965 ECHO REPLY
=====

```

Figura 5: ECHO y ECHO REPLY capturados en tiempo real

```

Snort processed 659 packets.
Snort ran for 0 days 0 hours 2 minutes 13 seconds
  Pkts/min:      329
  Pkts/sec:       4
=====
Packet I/O Totals:
  Received:      659
  Analyzed:      659 (100.000%)
  Dropped:       0 ( 0.000%)
  Filtered:      0 ( 0.000%)
  Outstanding:   0 ( 0.000%)
  Injected:      0
=====
Breakdown by protocol (includes rebuilt packets):
  Eth:           659 (100.000%)
  VLAN:          0 ( 0.000%)
  IP4:           625 ( 94.841%)
  Frag:          0 ( 0.000%)
  ICMP:          8 ( 1.214%)
  UDP:          102 ( 15.478%)
  TCP:          515 ( 78.149%)
  IP6:           10 ( 1.517%)
  IP6 Ext:       10 ( 1.517%)
  IP6 Opts:      0 ( 0.000%)
  Frag6:         0 ( 0.000%)
  ICMP6:         4 ( 0.607%)
  UDP6:         6 ( 0.910%)

```

Figura 6: Resumen paquetes detectados en escaneo a tiempo real

3.1.2.3. Snort como NIDS

Como se ha destacado anteriormente, una de las principales características de Snort es la posibilidad de tratar de una forma predefinida a aquellos paquetes que son de interés para quien implementa el NIDS. Esto puede implementarse con el uso de reglas, tanto propias como aquellas predefinidas por Snort.

Tras hacer los cambios descritos en el archivo *snort.conf*, el primer paso será testear con la opción “-T” la configuración del archivo en la interfaz de red que está escuchado Snort. Esta opción verifica que la configuración que se va a usar a continuación es válida y que no va a fallar al llevarse a cabo. Después de ejecutar el comando deberá mostrar un mensaje similar al de la Figura 7.

```
Snort successfully validated the configuration!
Snort exiting
c:\Snort\bin>
```

Figura 7: Validación de archivo de configuración

Una vez comprobado esto, se implementará una regla en el archivo `C:\Snort\rules\local.rules`. En este caso se pretende generar una alerta al detectar la presencia de un paquete ICMP en el tráfico capturado, sin importar su dirección IP o puerto de origen ni de destino. Además se añade un identificador, teniendo en cuenta la restricción de dejar reservados los números menores a 1000000.

```
alert icmp any any -> any any (msg:"ALERTA ICMP DE PRUEBA";
sid:1000001;)
```

Al relanzar Snort se cargará este archivo, que fue referenciado en el archivo de configuración, y se usará esa regla en todo el tráfico que pase por la interfaz en cuestión. Una vez más se captura en tiempo real, pero esta vez se añade la opción “-A” para activar el modo de alertas, “console” para que muestre las notificaciones por pantalla y “-q” para que no muestre más información que la que se desea en esta ocasión. Si mientras el escaneo se realiza un ping de nuestra máquina (IP 192.168.1.135) a otra con IP 192.168.1.130, aparecen paquetes ICMP y sus respectivas alertas. Como ilustra la Figura 8, aparecen ocho mensajes de alerta, uno por cada paquete enviado en el ping, junto con el SID que le fue asignado en el archivo de reglas locales.

```
snort -A console -q -c c:\Snort\etc\snort.conf -i 2
```

```
c:\Snort\bin>snort -A console -q -c c:\Snort\etc\snort.conf -i 2
01/27-17:24:02.582467 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.135 -> 192.168.1.130
01/27-17:24:03.096590 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.130 -> 192.168.1.135
01/27-17:24:03.592152 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.135 -> 192.168.1.130
01/27-17:24:03.601152 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.130 -> 192.168.1.135
01/27-17:24:04.601173 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.135 -> 192.168.1.130
01/27-17:24:04.607463 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.130 -> 192.168.1.135
01/27-17:24:05.610945 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.135 -> 192.168.1.130
01/27-17:24:05.616901 1:1000001:0 ALERTA ICMP DE PRUEBA 0 {ICMP} 192.168.1.130 -> 192.168.1.135
```

Figura 8: Alertas ante paquetes ICMP detectados

Además, Snort no solo ha mostrado por pantalla dichas alertas, si no que lleva un registro en el directorio `C:\Snort\log`, creando en cada escaneo un archivo `snort.log.xxxxxxxxxx`.

3.2. Herramientas en torno a Snort

El IDS Snort dispone de una serie de herramientas para complementar su función principal. Algunas de estas herramientas son IDS Policy Manager,

Cyberprobe, Snorby, BASE o PMGraph y el propósito de este trabajo es probarlas en el entorno descrito, o en su defecto sobre Ubuntu, si fuera viable.

3.2.1. IDS Policy Manager

Esta herramienta permite la configuración remota de políticas de Snort, con un intuitivo interfaz gráfico que simplifica mucho la configuración de reglas en forma de políticas. Es útil para hacer cambios rápidos y administrar múltiples sensores, con diferentes políticas, a través del fichero de configuración *snort.conf*.

El primer problema enfrentado es la descarga del software. Debido a su antigüedad, la página oficial del creador ha dejado de estar disponible y actualmente el único medio de descarga es de fuentes no oficiales (y por tanto no seguras), como usuarios que conservan los archivos necesarios para la instalación y lo han subido a plataformas de almacenamiento como Mega para compartirlo con otros usuarios. Además, ya que IDS Policy Manager fue diseñado para trabajar con Windows 2000 o XP, es incompatible con su uso en Windows 10, por lo que se ha instalado el sistema operativo Windows XP en su versión de 32 bits en una máquina virtual en VirtualBox para su implementación.

Ya que este software es un complemento de Snort, se debe realizar de nuevo la configuración completa de este programa principal en la máquina virtual, como se ha descrito anteriormente. Para verificar su correcto funcionamiento se capturan una serie de pings entre el anfitrión y la máquina virtual, y para ello deben de estar conectados a la misma red física. Con este objetivo, se configura el adaptador de red de la máquina virtual como adaptador puente y se le asigna la dirección IP estática 192.168.1.138, de modo que se realizarán pings de la dirección 192.168.1.135 a la 192.168.1.138 y viceversa para comprobar, por ejemplo, que se observan las alertas ICMP establecidas.

Completado este paso, se procede a instalar Microsoft .NET Framework, ya que es un prerequisite para instalar IDS Policy Manager, y a continuación se ejecuta el instalador descargado y se completa una instalación de una manera muy intuitiva. Antes de continuar, se debe obtener el “Oink Code”, que se trata de una clave única asociada a una cuenta de usuario de Snort que sirve para cargar los paquetes de reglas disponibles en www.snort.org. Ésta puede obtenerse en la página de configuración de la cuenta una vez el usuario se ha registrado, y debe introducirse en primer lugar en Options > Settings, en el IDS Policy Manager como muestra la Figura 9.[13]

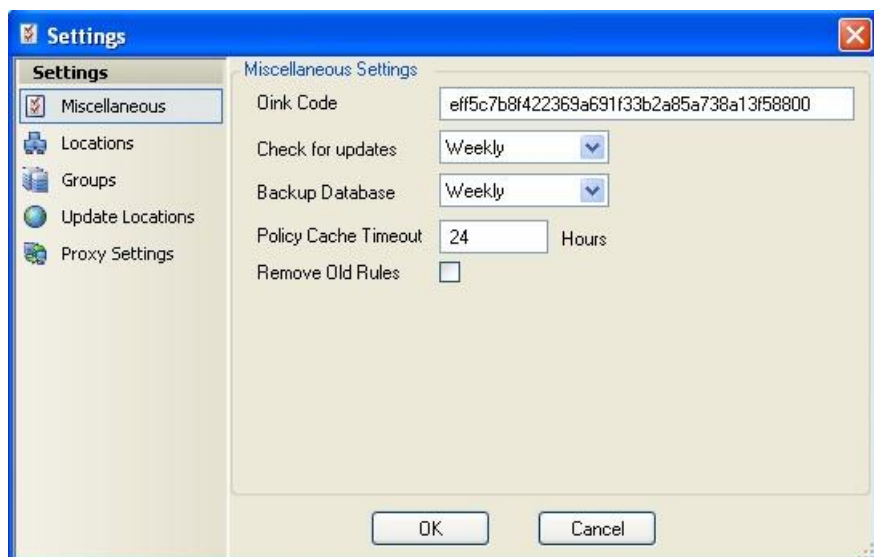


Figura 9: Oink Code en IDS Policy Manager

Tras introducir el Oink Code, se procede a la creación de una “policy”. Para ello se selecciona *Add Policy* sobre “Snort Policies” y se abre una pestaña en la que se indica el nombre de la política que se va a crear y la versión de Snort que se emplea. Al pulsar OK, aparece la ventana en el que se indica la ubicación del archivo de configuración snort.conf que se va a emplear. Todo ello se muestra en la Figura 10.

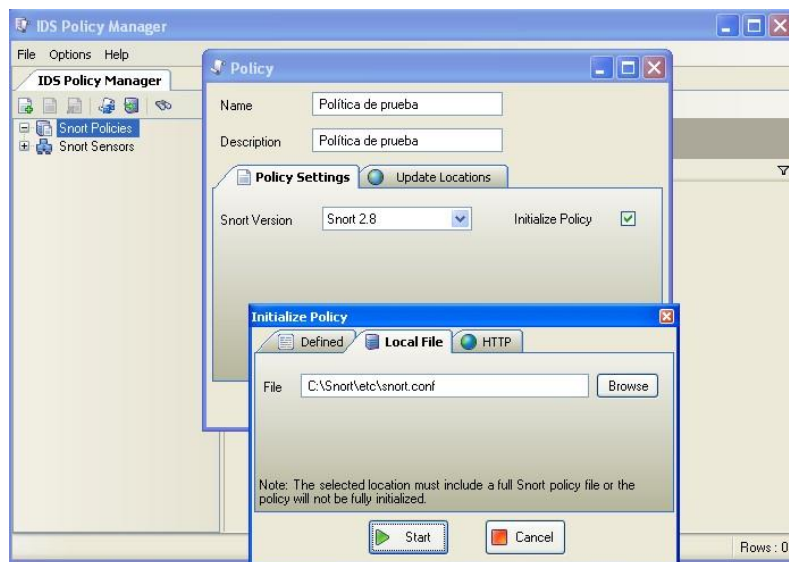


Figura 10: Creación de nueva política en IDS Policy Manager

Una vez se selecciona *Start*, se inicia el procedimiento de lectura del fichero de configuración y se crea la política “Política de prueba”, bajo la cual se puede desplegar un árbol que permite consultar y modificar las reglas dentro de esa política configurada.

El proceso de creación de una regla empleando IDS Policy Manager se realiza de una manera mucho más intuitiva gracias a su interfaz gráfica. En este caso, para generar una regla de alerta similar a la creada en Snort, basta con seleccionar “Add Item” sobre el grupo de reglas que interese y rellenar los datos necesarios, como muestra la Figura 11.

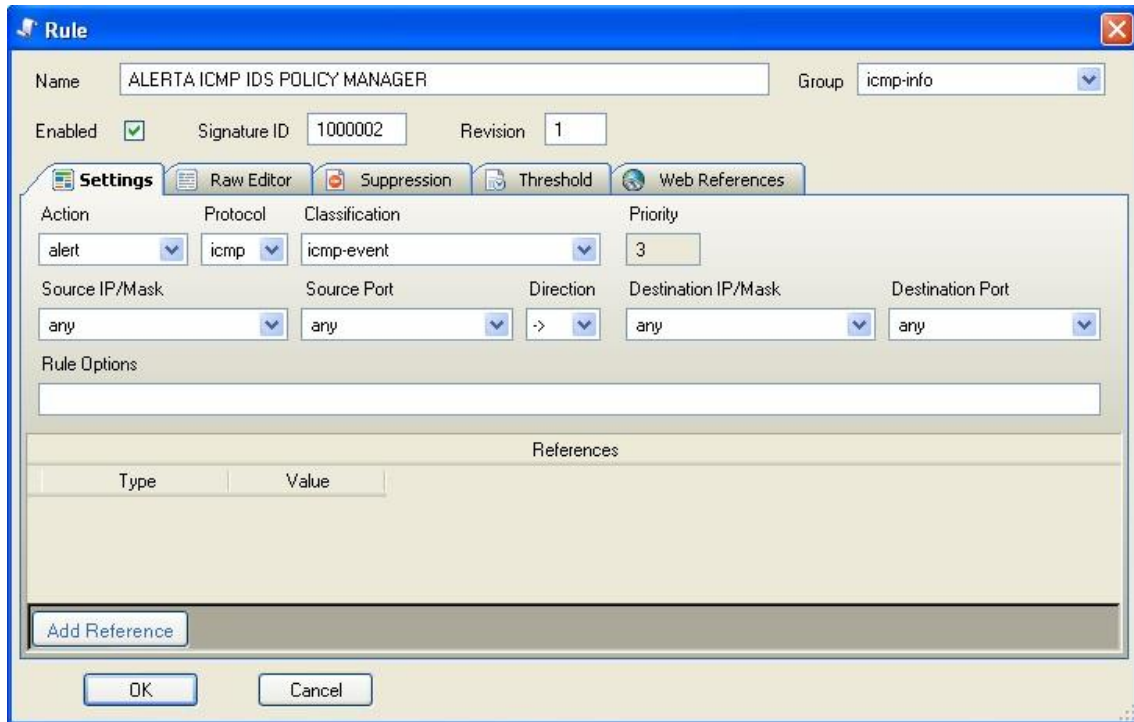


Figura 11: Creación de nueva alerta en IDS Policy Manager

De igual manera, IDS Policy Manager también permite crear sensores a los que aplicar políticas. Para ello una vez más basta con darle un nombre e indicar el directorio del archivo snort.conf en Snort Sensors > Add Sensor, como se muestra en la Figura 12.

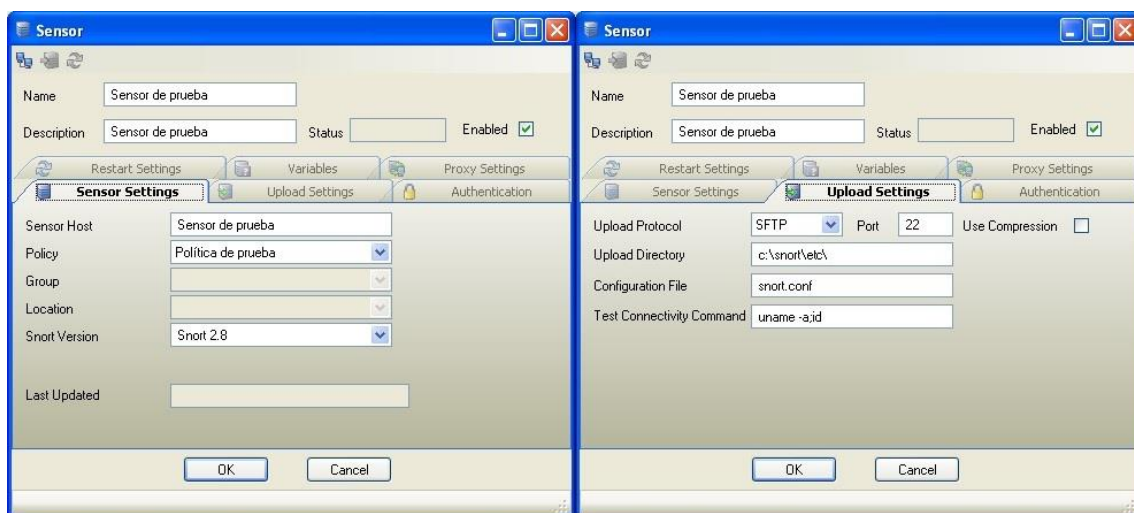


Figura 12: Creación de sensor en IDS Policy Manager

Por último, ya se pueden observar listados tanto el sensor como la política que se le ha aplicado, tal como reflejan las Figuras 13 y 14. Como se puede comprobar, se trata de una forma mucho más intuitiva de generar reglas.

SID	Enabled	Name	Action	Priority	Protocol	SrcIP	SrcPort	Direction	DstIP	DstPort
1000002	True	ALERTA ICMP IDS POLICY MANAGER	alert	3	icmp	any	any	->	any	any

Figura 13: Alerta creada en IDS Policy Manager

Enabled	Name	Policy Name	Status	Description
True	Sensor de prueba	Política de prueba	Not Current	Sensor de prueba

Figura 14: Sensor creado en IDS Policy Manager

3.2.2. Cyberprobe

La herramienta Cyberprobe se encarga de monitorizar redes contra ciberataques, y en combinación con Snort contrasta la dirección IP con los datos recogidos por éste. Este software está diseñado para plataformas Linux/Unix y consta de dos partes, que pueden utilizarse juntas o por separado:

- Cyberprobe: Sonda que recoge los paquetes de datos y los reenvía a través de una red con protocolos de transmisión estándar. Se puede configurar para recibir alertas de Snort de modo que cuando se reciba una, la dirección IP origen de ésta sea monitorizada de forma dinámica por un período de tiempo y así identificar a un potencial atacante. Opcionalmente, la sonda también puede ejecutar una interfaz de gestión que permite alterar de manera remota el mapa de focalización y la configuración.
- Cybermon: Monitor que recibe los paquetes de streaming, decodifica los protocolos a tiempo real e interpreta la información de la manera que el usuario haya definido con el lenguaje de programación Lua. Incluye técnicas de falsificación de paquetes, que permiten restablecer conexiones TCP y falsificar respuestas DNS con el fin de luchar contra los ataques de red, y soporta los protocolos IP, TCP, UDP, ICMP, HTTP y DNS.[11]

Debido a su diseño y aprovechando las anteriores pruebas en Ubuntu, se empleará este sistema operativo para la demostración de su funcionamiento, realizando la instalación con los comandos descritos en el archivo “README” para Linux. Para verificar la utilidad de esta herramienta, deben de configurarse tres parámetros:

- Interfaces: Para ello se creará un fichero de configuración llamado *c.cfg* que declare la interfaz de nuestra red, con el formato mostrado

en la Figura 15, y se pondrá en marcha con el comando “*sudo cyberprobe c.cfg*”, obteniendo por respuesta el mensaje “*Capture on interface enp0s8 started.*”, lo que indica que se puede pasar al siguiente paso.

- **Objetivos:** Una vez obtenido el mensaje de verificación de la interfaz, se define la red sobre la que se quiere actuar, añadiendo al fichero de configuración de la Figura 15 el apartado “targets”. Al instante, se detectará la nueva configuración definida y se muestra el mensaje “*Added target 192.168.56.107/32 -> 123456*”.
- **Endpoints:** Por último, se define un lugar a donde serán enviados los datos capturados, con el apartado “endpoints” de dicha figura. Antes, se ejecuta en un terminal nuevo el comando “*etsi-rcvr 10000 / tcpdump -n -r -*” para verificar que hay salida del tcpdump en el momento en el que se carga la configuración y hay movimiento de datos.[14]



```
{
  "interfaces": [
    {
      "interface": "enp0s8"
    }
  ],
  "targets": [
    {
      "address": "192.168.56.107",
      "device": "123456"
    }
  ],
  "endpoints": [
    {
      "hostname": "localhost",
      "port": 10000,
      "transport": "tcp",
      "type": "etsi"
    }
  ]
}
```

Figura 15: Archivo c.cfg de Cyberprobe

Tras estos tres pasos se ha obtenido una serie de respuestas por parte de Cyberprobe, como se refleja en la Figura 16.

```
naiara@naiara-VirtualBox:~/cyberprobe$ sudo cyberprobe c.cfg
Capture on interface enp0s8 started.
Added target 192.168.56.107/32 -> 123456.
Added endpoint localhost:10000 of type etsi with transport tcp
ETSI LI connection to localhost:10000 established.
█

naiara@naiara-VirtualBox: ~
Archivo Editar Ver Buscar Terminal Ayuda
naiara@naiara-VirtualBox:~$ sudo etsi-rcvr 10000 |tcpdump -n -r -
[sudo] contraseña para naiara:
reading from file -, link-type RAW (Raw IP)
Target 123456 discovered on IP 192.168.56.107
█
```

Figura 16: Mensajes de verificación de Cyberprobe

Llegado este punto y volviendo al objetivo principal de la implementación de Cyberprobe, se realiza la integración con Snort. Para ello se debe tener creada una regla en el archivo “*local.rules*”, como se ha realizado previamente, y después configurar Cyberprobe para recibir las alertas de Snort, incluyendo en el fichero de configuración la sección:

```
“snort-alerters”: [
    { “path”: “/var/log/snort/snort_alert”,
      “duration”: “60”
    }
]
```

Se recibirá una respuesta indicando “*Start snort alerter on /var/log/snort/snort_alert*” y, en este momento, se puede ejecutar snort en un terminal con el comando “*sudo snort -i enp0s8 -A unsock -N -j /var/log/snort/ -c /etc/snort/snort.conf*”, y en otro terminal simultáneamente se ejecuta Cyberprobe activado con el comando “*sudo cyberprobe c.cfg*”. En este escenario, una vez se produzca el suceso que haga saltar la alarma, se observa cómo Cyberprobe muestra la alerta que se ha definido, tal y como se observa en la Figura 17.


```

nailara@nailara-VirtualBox:~/snort_src/cyberprobe$ sudo cyberprobe c.cfg
Start snort alerter on /var/log/snort/snort_alert
Added endpoint localhost:10000 of type etsi with transport tcp
Added target 192.168.56.108/32 -> 123456.
Capture on interface enp0s8 started.
Snort alert: ALERTA ICMP CYBERPROBE
Hit on signature ID 1000001, targeting 192.168.56.1
Snort alert: ALERTA ICMP CYBERPROBE
Hit on signature ID 1000001, targeting 192.168.56.108
Stopped targeting on 192.168.56.1
Stopped targeting on 192.168.56.108

```

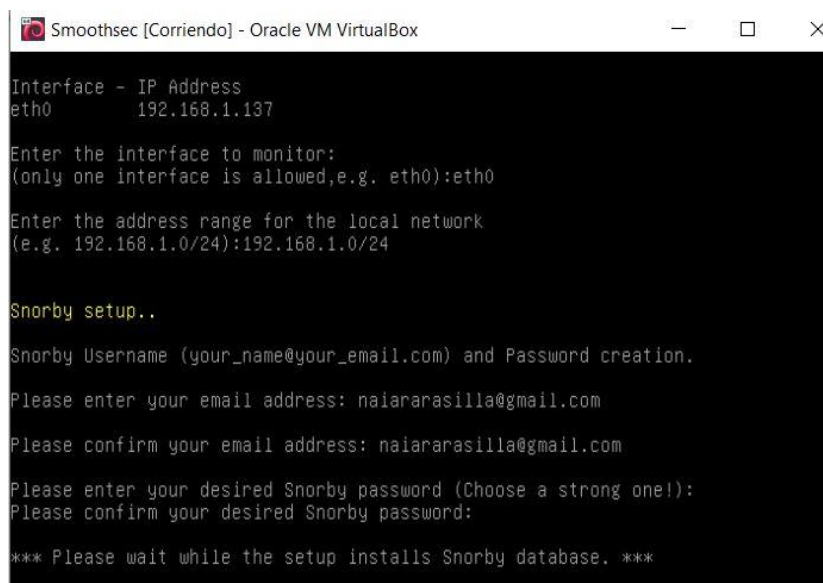
Figura 17: Alerta mostrada por Cyberprobe

Por último, una parte interesante de esta herramienta es Cybermon. Para trabajar con esta interfaz se ejecutaría el comando “*sudo cybermon -p 10000 -c /usr/local/etc/cyberprobe/monitor.lua*”. Sin embargo, debido a sus librerías desactualizadas no se ha logrado el acceso.

3.2.3. Snorby

Este módulo consiste en una aplicación web para gestión de alertas, que permite monitorizar mediante una interfaz gráfica la seguridad de la red en combinación con IDS como Snort, que genera eventos de log en el formato Unified2.

Para su instalación es útil ayudarse del instalador de Smoothsec, que es un sistema operativo compatible con Linux que da acceso al usuario a IDS como Snort o Suricata, y en el que siguiendo unos pasos establecidos por el instalador se puede configurar Snorby para trabajar en la red objetivo, como muestra la Figura 18.



```

Smoothsec [Corriendo] - Oracle VM VirtualBox
Interface - IP Address
eth0      192.168.1.137

Enter the interface to monitor:
(only one interface is allowed,e.g. eth0):eth0

Enter the address range for the local network
(e.g. 192.168.1.0/24):192.168.1.0/24

Snorby setup..

Snorby Username (your_name@your_email.com) and Password creation.

Please enter your email address: nailararasilla@gmail.com
Please confirm your email address: nailararasilla@gmail.com

Please enter your desired Snorby password (Choose a strong one!):
Please confirm your desired Snorby password:

*** Please wait while the setup installs Snorby database. ***

```

Figura 18: Configuración de parámetros Snorby

Una vez finalizada la configuración, se puede observar el banner de Smoothsec seguido de la dirección IP que ha sido asignada de manera estática, que será la que se emplea para acceder a la aplicación web de Snorby. En las Figuras 19 y 20 se muestra el aspecto del banner y la página de entrada a dicha herramienta en la que se introducen los datos dados en la instalación, respectivamente.



Figura 19: Dirección IP de la aplicación web

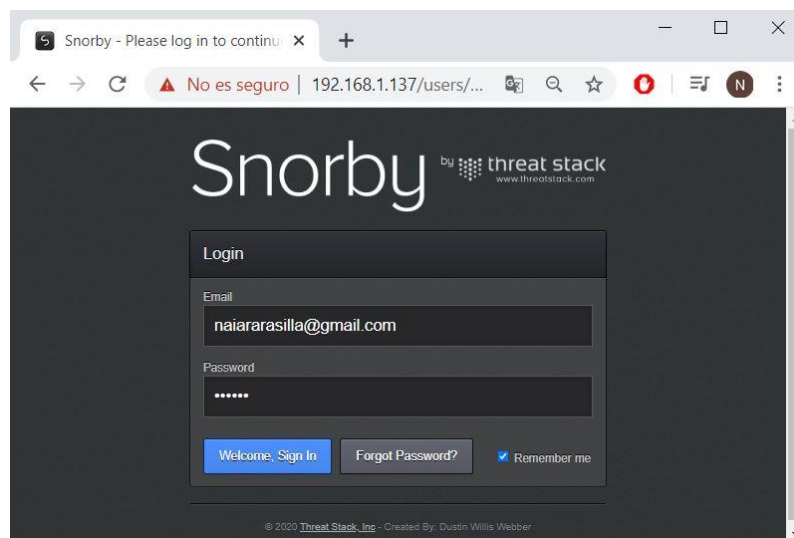


Figura 20: Entrada a Snorby

Una vez se ha accedido a la aplicación web, se observa el dashboard. Esto es un tablero de reportes que muestra los eventos registrados en cierto intervalo de tiempo, permitiendo filtrarlos por su grado de gravedad, su origen o destino, el protocolo empleado o las reglas asociadas. En la columna derecha se observan, a su vez, los elementos más recientes detectados y clasificados por su naturaleza. La Figura 21 ilustra el estado del dashboard tras hacer diferentes interacciones con la máquina monitorizada, que se pueden analizar más en detalle en la pestaña “Events”, lo cual se desarrolla en la Figura 22, en la que se ven los datos y protocolos que se detectan en cada uno.

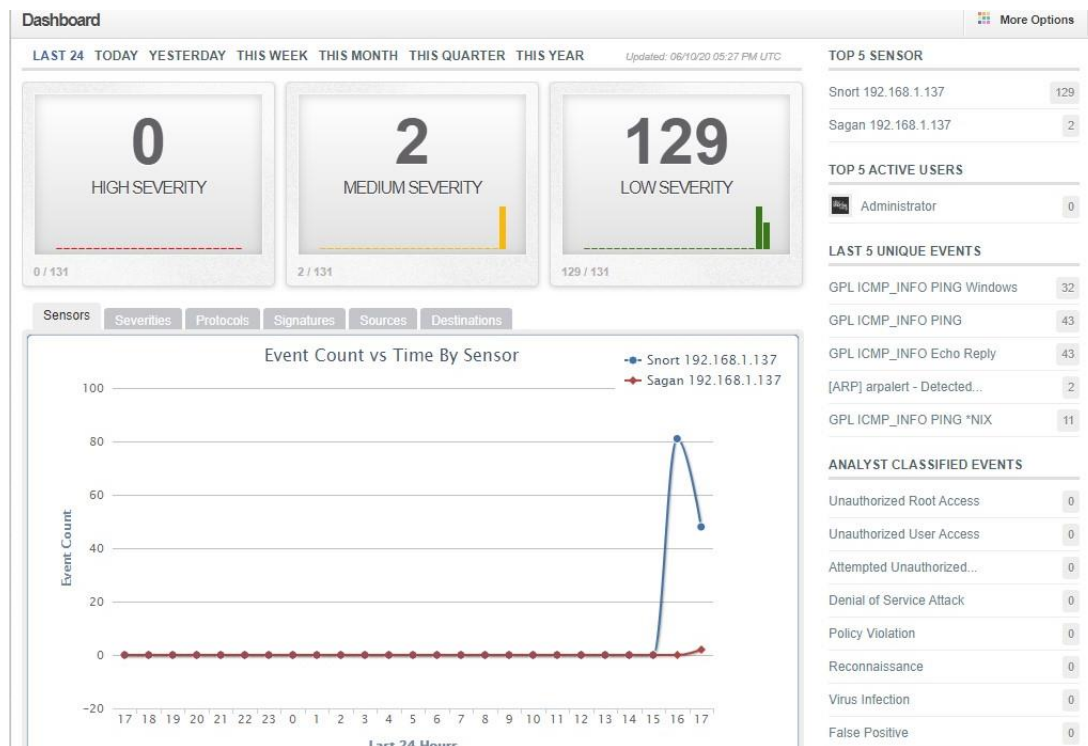


Figura 21: Dashboard Snorby

Snorby SPONSORED BY **threat stack** <https://threatstack.com> Welcome Administrator | Settings | Log out

Dashboard | My Queue (0) | Events | Sensors | Search | Administration

Listing Sessions (9 unique unclassified sessions) Hotkeys Classify Event(s) Filter Options

Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp	Sessions
3	Snort	192.168.1.129	192.168.1.137	GPL ICMP_INFO PING Windows	5:18 PM	24

IP Header Information View All Sessions Perform Mass Classification Event Export Options Permalink

Source	Destination	Ver	Hlen	Tos	Len	ID	Flags	Off	TTL	Proto	Csum
192.168.1.129	192.168.1.137	4	5	0	60	28102	0	0	128	1	18592

Signature Information

Generator ID	Sig. ID	Sig. Revision	Activity (32/131)	Category	Sig Info
1	2100382	8	24.43%	misc-activity	Query Signature Database View Rule

ICMP Header Information

Type	Code	Csum	ID	SEQ
8	0	19614	1	189

References

Type	Value
arachnids	http://www.whitehats.com/info/IDS169

Payload Hex Ascii

```

000000: 08 00 27 de 88 39 28 39 26 52 41 d1 08 00 45 00 00 3c 6d c6 00 00 80 01 48 a0 ..'.9(98RA...E...cm.....H.
000010: c0 a8 01 81 c0 a8 01 89 08 00 4c 9e 00 01 00 bd 61 62 63 64 65 66 67 68 69 6a .....L.....abcdefghij
000020: 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69 klmnopqrstuvwxyzabcdefghi

```

Notes

This event currently has zero notes - You can add a note by clicking the button below.

Add A Note To This Event

3	Snort	192.168.1.129	192.168.1.137	GPL ICMP_INFO PING	5:18 PM	35
3	Snort	192.168.1.137	192.168.1.129	GPL ICMP_INFO Echo Reply	5:18 PM	35
2	Sagan	192.168.1.133	192.168.1.137	[ARP] arpalert - Detected new machine on the network	5:04 PM	1
3	Snort	192.168.1.138	192.168.1.137	GPL ICMP_INFO PING Windows	5:01 PM	8
3	Snort	192.168.1.138	192.168.1.137	GPL ICMP_INFO PING	5:01 PM	8
3	Snort	192.168.1.137	192.168.1.138	GPL ICMP_INFO Echo Reply	5:01 PM	8
2	Sagan	192.168.1.138	192.168.1.137	[ARP] arpalert - Detected new machine on the network	5:01 PM	1
3	Snort	192.168.1.129	192.168.1.137	GPL ICMP_INFO PING *NIX	4:59 PM	11

Figura 22: Eventos en Snorby

Se observa que se han realizado una serie de pings desde las direcciones IP 192.168.1.129, 192.168.1.138 y 192.168.1.133, que son máquinas físicas o virtuales conectadas a la misma red, y se les da el color verde porque se consideran detecciones con poco grado de peligrosidad. En el caso de las dos sesiones de eventos marcadas de amarillo, se consideran de peligrosidad media, ya que se trata de detecciones de estas máquinas que se acaban de meter en la red, y no siempre los accesos a una red son deseados.

También son útiles las pestañas “My Queue” en la que se puede agregar un evento sospechoso para llevar su seguimiento de una forma más aislada y la pestaña “Search”, para personalizar la búsqueda de detecciones (por

ejemplo, las relacionadas solo con una dirección IP, o con un sensor en el caso de monitorizar más de una máquina). Además, volviendo a la pestaña del dashboard, se pueden extraer datos más visuales sobre las detecciones, véase los diagramas de sectores como el de la Figura 23 en el que se representa el porcentaje de detecciones según su origen, y si se desea puede extraerse un informe completo en formato pdf de todos estos aspectos haciendo clic sobre *More options* > *Export to PDF*.

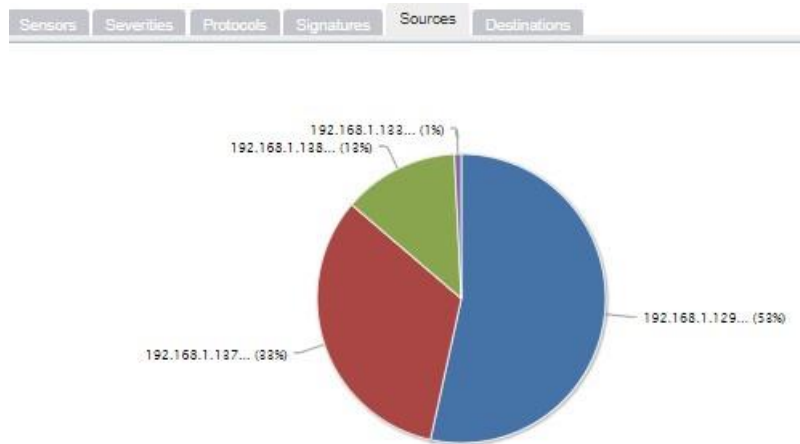


Figura 23: Diagrama origen detecciones Snorby

3.2.4. BASE y PMGraph

El proyecto Basic Analysis and Security Engine, también conocido como BASE, es una herramienta para monitorizar la seguridad en la red con una interfaz web, atendiendo alertas de un IDS como Snort.

Para su instalación, al igual que se hizo con Snorby, puede emplearse un paquete que incluye varias herramientas, llamado EasyIDS. Se ha elegido esta forma porque a través de una misma máquina virtual se puede acceder a BASE y PMGraph, que se trata de otra herramienta de visualización para auditar la red.

Una vez instalada la máquina virtual y configurados los parámetros, aparece el mensaje de bienvenida de EasyIDS junto con la dirección IP asignada, para acceder en este caso a <http://192.168.1.139>, como se deduce de la Figura 24. Una vez se ha accedido a esa dirección, se pide modificar la clave que se proporciona por defecto (password) para hacerlo seguro. En la Figura 25 se muestra el paso de cambio de clave, en el que se introduce una clave robusta.

```

Welcome to EasyIDS
-----

For access to the EasyIDS web GUI use this URL
https://192.168.1.139

For help on EasyIDS commands you can use from this
command shell type help-easyids.

[root@easyids ~]# _

```

Figura 24: Mensaje bienvenida EasyIDS

Change Passwords

MySQL Root password

For security reasons you must change the MySQL root password!!!

Password: Confirm:

Figura 25: Cambio de clave en EasyIDS

Por otro lado, en la Figura 26 se observan los servicios que están activados de aquellos que integran EasyIDS.

System Status	
Services	
Arpwatch	RUNNING
Barnyard	RUNNING
CRON server	RUNNING
MySQL	RUNNING
NTOP	RUNNING
NTP Server	RUNNING
Secure shell	RUNNING
Snort	RUNNING
Stunnel	RUNNING
Web server	RUNNING

Figura 26: Servicios activos en EasyIDS

Para trabajar con esta herramienta como un detector de intrusiones es conveniente generar un fichero de alertas en el directorio /etc/snort/rules, en el que se define en este caso una alerta ante la presencia de paquetes con el protocolo ICMP, con el siguiente formato:

```

alert icmp any any -> any any (msg:"AVISO, ICMP detectado!!";
sid:1000001;)

```

Ahora, se ejecuta Snort en modo sniffer como previamente y al realizar pings desde cualquier máquina se observa que salta la alerta de la manera tradicional. El objetivo de BASE es poder analizar la actividad monitorizada por Snort de una manera mucho más visual, por ello en la Figura 27 indica los protocolos que se han detectado en el escaneo y, por ejemplo, se puede hacer un listado de las alertas que han saltado en un día y ver sus detalles, como ilustran la Figura 28 y 29, respectivamente.

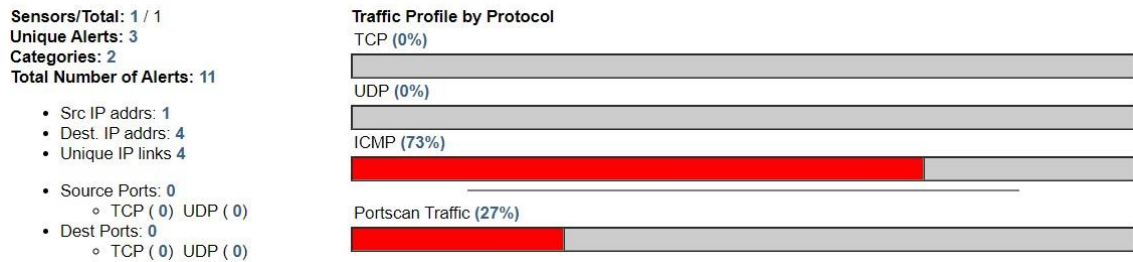


Figura 27: Protocolos detectados en BASE

<input type="checkbox"/>	ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
<input checked="" type="checkbox"/>	#0-(1-1)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:23:38	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#1-(1-2)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:23:43	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#2-(1-3)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:23:48	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#3-(1-4)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:23:53	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#4-(1-5)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:24:07	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#6-(1-6)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:24:12	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#6-(1-7)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:24:17	192.168.1.132	192.168.1.139	ICMP
<input checked="" type="checkbox"/>	#7-(1-8)	[snort] Snort Alert [1:1000001:0]	2020-06-22 16:24:22	192.168.1.132	192.168.1.139	ICMP
<input type="checkbox"/>	#8-(1-9)	[snort] portscan: TCP Portsweep	2020-06-22 16:25:11	192.168.1.132	52.114.128.9	Raw IP
<input type="checkbox"/>	#9-(1-10)	[snort] portscan: Open Port	2020-06-22 16:25:13	192.168.1.132	52.109.76.8	Raw IP
<input type="checkbox"/>	#10-(1-11)	[snort] portscan: Open Port	2020-06-22 16:25:13	192.168.1.132	104.126.105.221	Raw IP

Figura 28: Alertas detectadas en BASE

Meta	ID #		Time		Triggered Signature								
	1 - 2		2020-06-22 16:23:43		[snort] Snort Alert [1:1000001:0]								
	Sensor	Sensor Address		Interface		Filter							
		easysids		eth1		none							
	Alert Group		none										
IP	Source Address		Dest. Address		Ver	Hdr Len	TOS	length	ID	fragment	offset	TTL	chksum
	192.168.1.132		192.168.1.139		4	20	0	60	26919	no	0	128	19770 = 0x4d3a
	Options		none										
ICMP	type		code	checksum	ID	seq #							
	(8) Echo Request		(0) 0	19706 = 0x4cfa	1	97							
Payload	length = 32												
	000 : 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 abcdefghijklmnop												
	010 : 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69 qrstuvwabodeefghi												

Figura 29: Detalles de alerta en BASE

Por último, BASE también muestra gráficamente estadísticas sobre los datos recogidos. Se puede representar en forma de diagrama de sectores, de barras o de manera lineal, y además se puede seleccionar el intervalo de tiempo para mostrar solo los datos que interesen.

Por ejemplo, para realizar el gráfico de la Figura 30 se ha elegido representar el número de alertas que se ha detectado de cada origen, generando alertas desde varias máquinas tanto reales como virtuales, aunque pueden realizarse multitud de clasificaciones. El gráfico refleja que se han detectado con Snort once alertas procedentes de la máquina con IP 192.168.1.132, tres alertas desde la 192.168.1.133 y diez con origen en la IP 192.168.1.139.

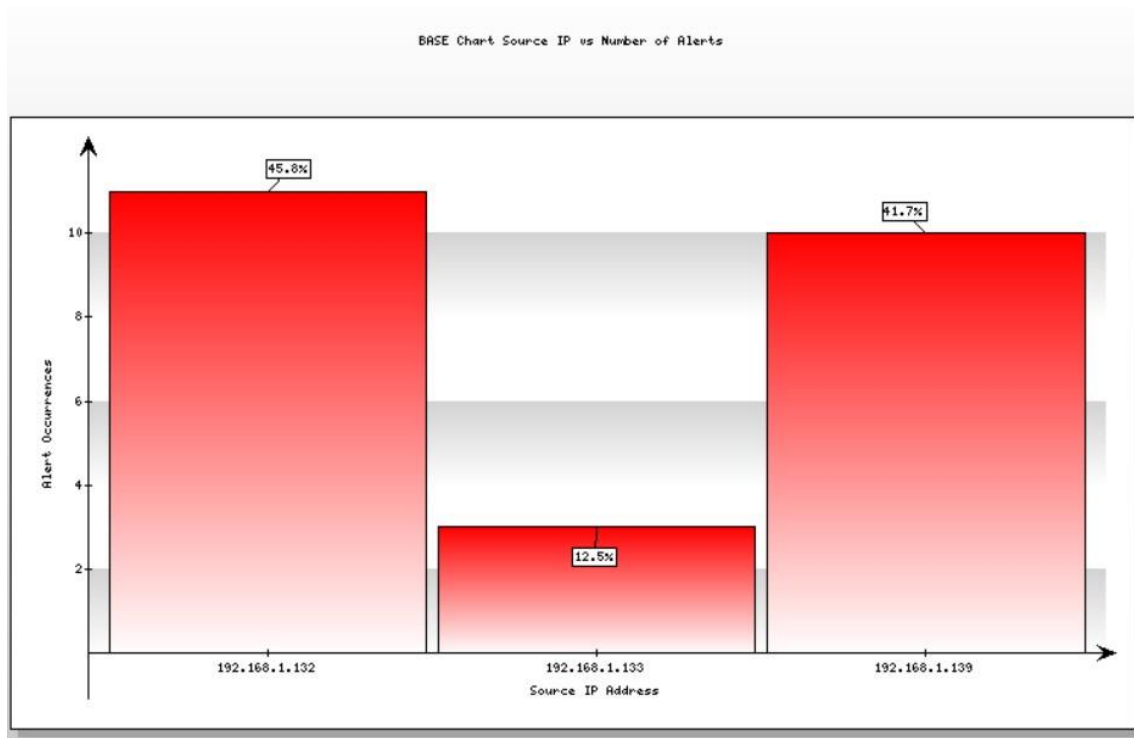


Figura 30: Gráfico alertas en BASE

Para concluir con esta máquina virtual con EasyIDS, se muestra en la Figura 31 una gráfica que crea la herramienta PMGraph, en la que se expresa el tráfico de la interfaz que se ha analizado en Bytes por segundo. Asimismo, pueden obtenerse gráficos similares con datos como el número de paquetes detectados por Snort, las alertas por segundo o el uso de recursos como la CPU o la memoria. Para acceder a todos estos esquemas se elige la sección "Graphs".

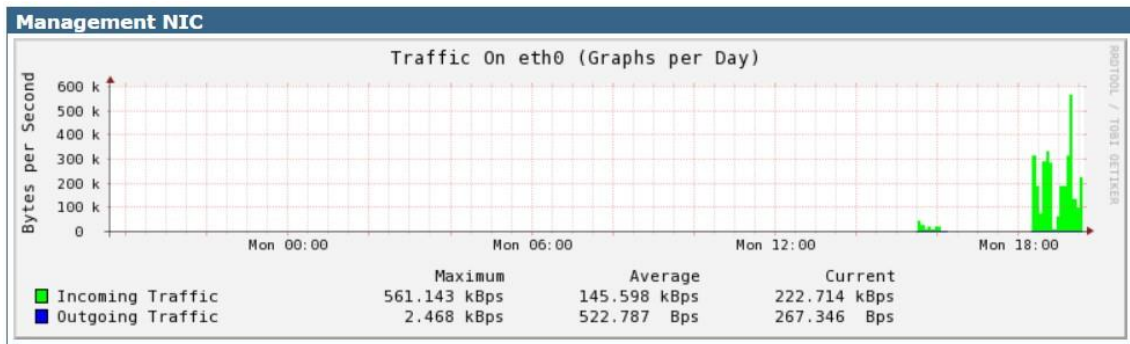


Figura 31: Tráfico en eth0 con PMGraph

4. Ejemplo de aplicación

Para comprobar el funcionamiento en un entorno que se asemeja más a la realidad conviene realizar una intrusión real al sistema y probar cómo se comporta éste cuando no tiene ninguna protección de un IDS, en comparación a cuando está Snort en funcionamiento.

4.1. Plataforma desplegada

La máquina atacante es una VM Kali Linux, ya que permite utilizar la herramienta Metasploit, que es una herramienta muy completa para desarrollar y ejecutar exploits contra una máquina remota. Las pruebas de intrusión o ataques se realizan a sistemas físicos o virtuales en los que previamente se ha instalado y configurado Snort para instalar alguna de las herramientas anteriores y, de este modo, complementar los ataques con registros gráficos de éstos.

4.2. Pruebas de intrusión

4.2.1. Ataque por escaneo de puertos

El “port scanning” o escaneo de puertos consiste en analizar mediante un programa el estado de los puertos de una máquina conectada a una red para detectar si un puerto está abierto, cerrado o protegido por un cortafuegos. Esta información sirve para detectar qué servicios tiene activos la máquina y tomar ventaja de ellos, y se puede realizar combinando Nmap y Metasploit.

Se comprueba su utilidad, en primer lugar, en un entorno desprotegido, y por último en uno más seguro, ejecutando el IDS Snort. El primer paso es cargar la base de datos y ejecutar la interfaz de Metasploit con los comandos *msfconsole* y *sudo msfdb run*. A continuación puede comprobarse cómo en la Figura 32 se muestra el banner de Metasploit y se comprueba que se ha conectado a la base de datos con el comando *db_status*. [15]

```
Metasploit

=[ metasploit v5.0.95-dev ]
+ -- --=[ 2038 exploits - 1103 auxiliary - 344 post ]
+ -- --=[ 562 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

Metasploit tip: After running db_nmap, be sure to check out the result of h
osts and services

msf5 > db_status
[*] Connected to msf. Connection type: postgresql.
```

Figura 32: Banner bienvenida a Metasploit

Acto seguido ya puede ejecutarse NMAP desde la consola de Metasploit para escanear la dirección IP del anfitrión. A continuación, se muestra un resumen de la respuesta por comandos, haciendo énfasis en la Figura 33, que contiene los datos relevantes en cuanto a puertos abiertos detectados y sistema operativo detectado, sin la protección de ningún sistema de detección de intrusiones.

```
msf5 > db_nmap -v -sV 192.168.1.132

[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2020-07-02 17:42 CEST
[*] Nmap: NSE: Loaded 45 scripts for scanning.
[*] Nmap: Initiating ARP Ping Scan at 17:42
[*] Nmap: Scanning 192.168.1.132 [1 port]

...

[*] Nmap: Discovered open port 135/tcp on 192.168.1.132
[*] Nmap: Discovered open port 445/tcp on 192.168.1.132
[*] Nmap: Discovered open port 139/tcp on 192.168.1.132
```

```
[*] Nmap: Nmap scan report for 192.168.1.132
[*] Nmap: Host is up (0.00017s latency).
[*] Nmap: Not shown: 997 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
[*] Nmap: 135/tcp open  msrpc        Microsoft Windows RPC
[*] Nmap: 139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
[*] Nmap: 445/tcp open  microsoft-ds?
[*] Nmap: MAC Address: 28:39:26:52:41:D1 (CyberTAN Technology)
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figura 33: Port scanning sin Snort

Si la máquina objetivo está ejecutando Snort, éste es capaz de avisar al detectar el escaneo de puertos gracias al pre-procesador que lleva integrado. En la Figura 34 se demuestra, al ejecutar Snort en la máquina “atacada” con

el comando “*snort -A console -i 1 -c C:/Snort/etc/snort.conf*” y lanzar el escaneo de puertos desde Kali, cómo Snort alerta al usuario de la detección.

```
Commencing packet processing (pid=8568)
07/02-19:15:50.224238  [**] [122:3:1] (portscan) TCP PortswEEP [**] [Classification: Attempted
Information Leak] [Priority: 2] {PROTO:255} 192.168.1.132 -> 192.168.1.135
```

Figura 34: Alerta escaneo de puertos Snort

Además, en el resumen mostrado al finalizar el análisis de Snort indica que se han detectado 2908 paquetes TCP en apenas dos minutos, por lo que otra forma de detectarlo sería incluir en el archivo *local.rules* una regla de detección de paquetes TCP entrantes a la máquina que se desea cubrir. Sin embargo, por su volumen de paquetes y teniendo la ventaja de que esta versión de Snort ya tiene la alerta integrada, es más conveniente prescindir de este método para no saturar la pantalla del usuario de alertas repetidas.

Si se desea una visualización más interactiva de los resultados, como la de la Figura 35, puede emplearse la herramienta Snorby. En la imagen se observa el registro de los intentos continuados por escanear puertos, que figuran como “Port unreachable”

<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:55 PM
<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:55 PM
<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:55 PM
<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:54 PM
<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:54 PM
<input type="checkbox"/>	★	3	Snort	192.168.1.137	192.168.1.1	GPL ICMP_INFO Destination Unreachable Port Unreachable	9:54 PM

Figura 35: Registro de escaneo de puertos en Snorby

4.2.2. Ataque por fuerza bruta

El ataque por fuerza bruta consiste en un método matemático, cuya dificultad depende de la cantidad de soluciones posibles, y es una forma de lograr conseguir la combinación usuario/contraseña correcta por parte de un intruso.

En internet se pueden encontrar numerosos diccionarios con los nombres de usuario y claves más usados. Algunos de ellos se han elaborado a partir de filtraciones de información que se han realizado en la historia de la informática, pero otros incluyen una búsqueda más personalizada, como por ejemplo usuarios con nombres propios típicos de España o usuarios típicos para routers según su fabricante. Esta segunda opción es la empleada para ilustrar este ataque por fuerza bruta, ya que es muy común entre usuarios no

avanzados el hecho de no cambiar las credenciales de acceso por defecto del router y, por lo tanto, es muy probable encontrarlas en un diccionario.

Para llevar a cabo este ataque basta con descargar dichos diccionarios de usuarios y contraseñas y emplear la herramienta *hydra*, que se trata de una de las aplicaciones más conocidas para crackear contraseñas. Puede ejecutarse por terminal, pero la máquina virtual Kali dispone de una versión gráfica, *xHydra*, en la que se seleccionan parámetros como la IP que se quiere atacar, las combinaciones de usuario y contraseña que se van a probar y el protocolo, y se genera un comando que se ejecuta desde su propia consola con un solo clic. En este caso, se realiza un intento de acceso SSH por cada combinación de usuario y contraseña, y al detectar una pareja de credenciales que permita el acceso a esa dirección IP avisa al usuario del descubrimiento, como representa la Figura 36. Finalmente, se puede acceder al terminal de la máquina atacada de forma remota mediante el comando “*ssh admin@192.168.1.1*”.

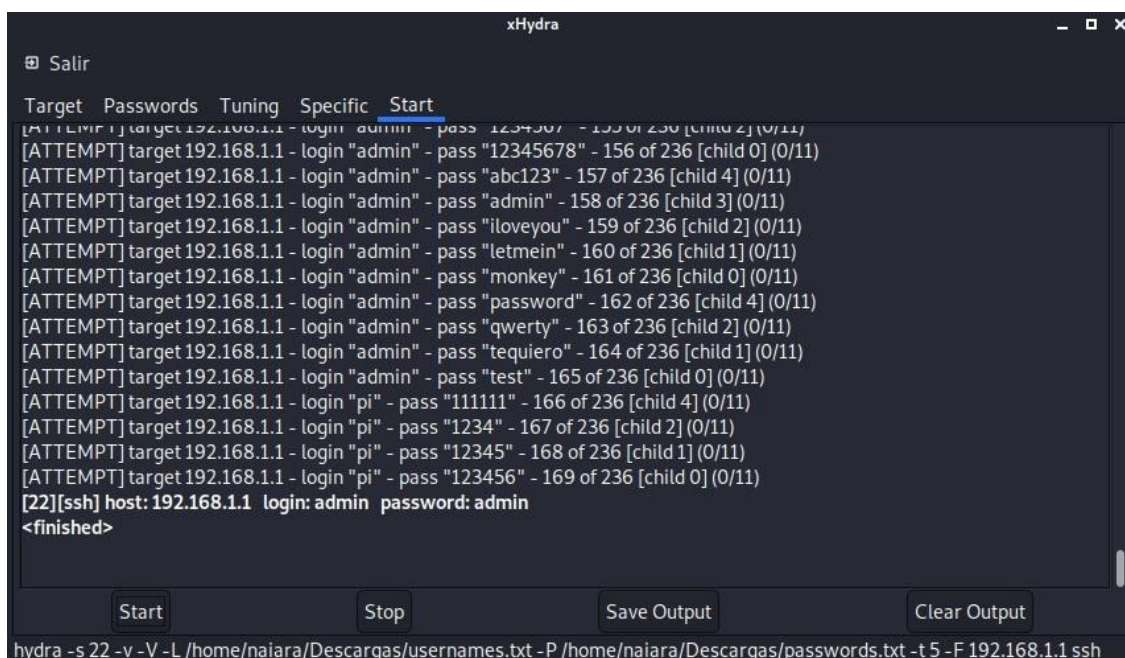


Figura 36: Credenciales descubiertas por xHydra

Snort puede intervenir en caso de detección de un ataque por fuerza bruta, pero previamente es necesario definir la siguiente alerta que detecte esta circunstancia:

alert tcp any any -> \$HOME_NET 22 (msg: "Posible ataque fuerza bruta SSH"; threshold: type both, track by_src, count 6, seconds 30; sid: 1000008;)

Definiendo para ello en el archivo de configuración la IP protegida como HOME_NET, e indicando así que el ataque SSH se realiza por el puerto 22,

y solicitando una alerta por pantalla cuando se detecten al menos seis intentos de acceso en 30 segundos, es decir, seis comprobaciones de si unas duplas usuario/contraseña son correctas. Para capturar la Figura 37 se lanza un ataque por fuerza bruta desde el atacante en Kali al equipo en el que ya se está ejecutando Snort y, en cuanto se detectan seis intentos de acceso, salta la alerta mostrada.[16]

```
C:\Snort\bin>snort -A console -q -i 1 -c C:\Snort\etc\snort.conf
07/09-20:45:04.267112  [**] [1:1000008:0] Posible ataque fuerza bruta SSH [**] [Priority: 0] {TCP}
192.168.1.136:47232 -> 192.168.1.134:22
```

Figura 37: Alerta ataque por fuerza bruta en Snort

Por último, para complementar esta detección, puede utilizarse la herramienta BASE para comprobar, como muestra la Figura 38, que han tenido lugar más de 90 alertas sobre intentos de ataque por fuerza bruta al puerto 22 TCP en esa hora.

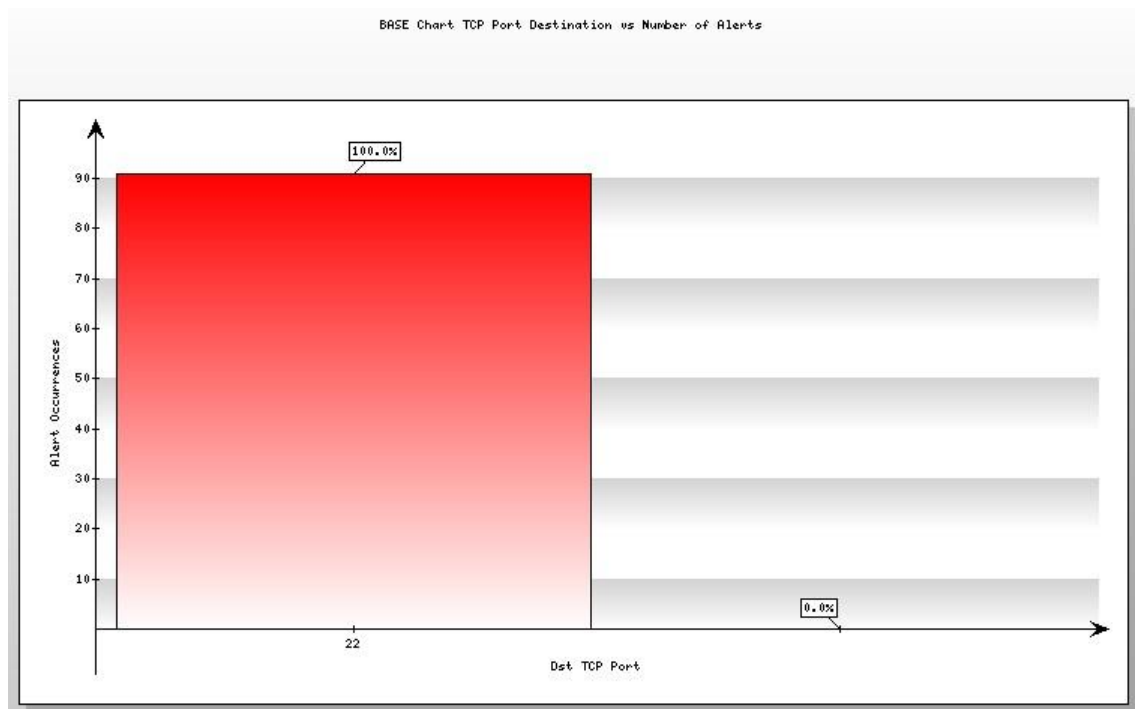


Figura 38: Gráfico alertas ataque fuerza bruta en BASE

4.2.3. Ataque DoS

En un ataque de Denegación de Servicio, el atacante genera una cantidad masiva de peticiones al servicio de una misma máquina o dirección IP con la intención de consumir sus recursos y dejarla sin capacidad de respuesta. En ese momento, la máquina se ve obligada a rechazar las peticiones, lo que se conoce como denegación de servicio. También existe la modalidad DDoS, en la que el atacante utiliza un gran número de máquinas o direcciones IP para concentrarse en un objetivo común y, por tanto, es más difícil de

bloquear la IP del atacante, aunque en esta ocasión basta con un DoS para comprobar el funcionamiento del proceso de ataque.

En un establecimiento de conexión TCP corriente se realiza el saludo a tres vías o 3-way handshake, en el que se envía un paquete SYN, se responde SYN/ACK y se confirma con ACK. Sin embargo, haciendo uso de la herramienta *hping3*, es posible modificar los paquetes que se envían a través del protocolo TCP/IP en función de las necesidades del atacante. De esta manera, el ataque de denegación de servicio se llevaría a cabo como refleja la Figura 39, obviando el envío del ACK final y enviando un paquete SYN tras otro al objetivo.[17][18]

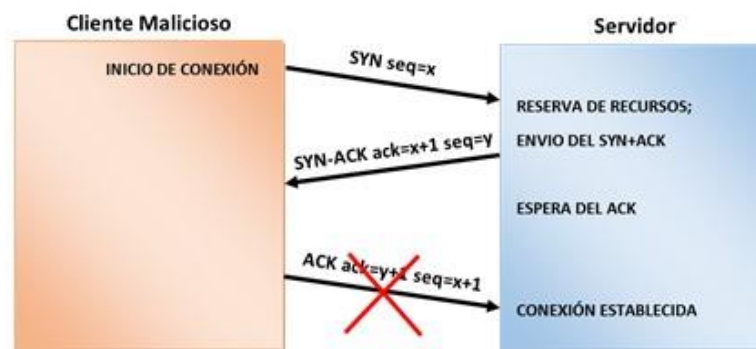


Figura 39: Esquema ataque DoS simple

Esta modalidad de ataque se implementará con un mero fin instructivo desde la máquina virtual Kali a una con Ubuntu, ya que no se pretende ocultar la identidad del atacante frente al receptor de la intrusión. Para ello se ejecuta desde Kali el comando “*hping3 -i u1 -S -p 80 192.168.1.138*” y a su vez se analizan con Wireshark los paquetes intercambiados. Es entonces cuando se captura la Figura 40, en la que se observa que cada microsegundo el atacante ha enviado un paquete del tipo SYN y el objetivo responde con un SYN/ACK, aunque se le van acumulando paquetes sin responder ya que no lo hace a la velocidad del atacante. Tras ese bombardeo de peticiones por parte de la máquina Kali y al encontrarse el atacado siempre consumiendo sus recursos y en espera de un ACK sin liberarlos, llega un instante en el se produce la denegación del servicio y la IP deja de responder a otras peticiones, aunque éstas sean legítimas.[17][18]

80602	2.635846416	192.168.1.137	192.168.1.138	TCP	56 29260 → 80 [SYN] Seq=0 Win=512 Len=0
80603	2.635935986	192.168.1.137	192.168.1.138	TCP	56 29261 → 80 [SYN] Seq=0 Win=512 Len=0
80604	2.636069163	192.168.1.138	192.168.1.137	TCP	62 80 → 29260 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...
80605	2.636077022	192.168.1.137	192.168.1.138	TCP	56 29260 → 80 [RST] Seq=1 Win=0 Len=0
80606	2.636069213	192.168.1.138	192.168.1.137	TCP	62 80 → 29261 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...
80607	2.636087092	192.168.1.137	192.168.1.138	TCP	56 29261 → 80 [RST] Seq=1 Win=0 Len=0
80608	2.636146069	192.168.1.137	192.168.1.138	TCP	56 29262 → 80 [SYN] Seq=0 Win=512 Len=0
80609	2.636162519	192.168.1.137	192.168.1.138	TCP	56 29263 → 80 [SYN] Seq=0 Win=512 Len=0
80610	2.636282029	192.168.1.138	192.168.1.137	TCP	62 80 → 29262 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...
80611	2.636288907	192.168.1.137	192.168.1.138	TCP	56 29262 → 80 [RST] Seq=1 Win=0 Len=0
80612	2.636282058	192.168.1.138	192.168.1.137	TCP	62 80 → 29263 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...
80613	2.636297788	192.168.1.137	192.168.1.138	TCP	56 29263 → 80 [RST] Seq=1 Win=0 Len=0
80614	2.636340362	192.168.1.137	192.168.1.138	TCP	56 29264 → 80 [SYN] Seq=0 Win=512 Len=0
80615	2.636467619	192.168.1.137	192.168.1.138	TCP	56 29265 → 80 [SYN] Seq=0 Win=512 Len=0
80616	2.636612015	192.168.1.138	192.168.1.137	TCP	62 80 → 29264 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...

Figura 40: Intercambio de paquetes en ataque DoS

Por otro lado, en una situación más afín a la realidad, el atacante tendría interés en ocultar su identidad para prevenir que le detengan su intento de DoS con un simple bloqueo a su dirección IP. Para ello se emplea un comando aleatorizador de la dirección IP de origen de esos paquetes SYN y, como muestra la Figura 41, el servidor responde a varias direcciones IP ajenas a la del culpable del ataque. Empleando Wireshark una vez más, como se puede observar en la Figura 42, tras añadir al comando anterior la extensión “--rand-source”, al analizar los paquetes hay múltiples direcciones IP de origen, a las que va respondiendo el atacado.

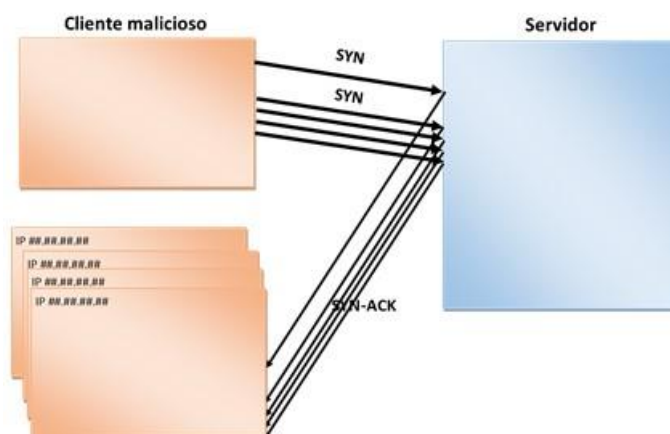


Figura 41: Esquema ataque DoS múltiples IPs

6066...	19.242781661	245.221.75.255	192.168.1.138	TCP	62 18060 → 80 [SYN] Seq=0 Win=512 Len=0
6066...	19.242857735	192.168.1.138	245.221.75.255	TCP	60 80 → 18060 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6066...	19.242983837	178.150.83.148	192.168.1.138	TCP	62 18061 → 80 [SYN] Seq=0 Win=512 Len=0
6066...	19.243031603	192.168.1.138	178.150.83.148	TCP	60 80 → 18061 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
6066...	19.243049072	45.73.107.9	192.168.1.138	TCP	62 18062 → 80 [SYN] Seq=0 Win=512 Len=0
6066...	19.243066592	192.168.1.138	45.73.107.9	TCP	60 80 → 18062 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

Figura 42: Intercambio de paquetes en ataque DoS aleatorizado

Como medida de protección ante este tipo de ataque, Snort permite una vez más crear reglas para detectarlo y alertar al usuario. La clave para definir esa regla es el gran número de repeticiones de envíos de paquetes con el protocolo TCP que se realizan en apenas un segundo. Por ello se define una regla que salte al detectar 100 paquetes TCP hacia el puerto 80 en 5 segundos:

```
alert tcp any any -> $HOME_NET 80 (msg:"Posible ataque DoS";  
threshold:type both, track by_src, count 100, seconds 5; sid:1000018;)
```

Al lanzar el ataque DoS no tarda en aparecer la alerta por parte de Snort. Además, puede emplearse la herramienta BASE para ver de manera muy visual como se han detectado gran cantidad de alertas dirigidas al puerto 80 TCP, como refleja la Figura 43 y 44, respectivamente.

```
C:\Snort\bin>snort -A console -q -i 1 -c C:\Snort\etc\snort.conf  
07/13-20:09:24.919317  [**] [1:1000018:0] Posible ataque DoS [**] [Priority: 0] {TCP} 192.168.1.137:61678 -> 192.168.1.135:80
```

Figura 43: Alerta DoS en Snort

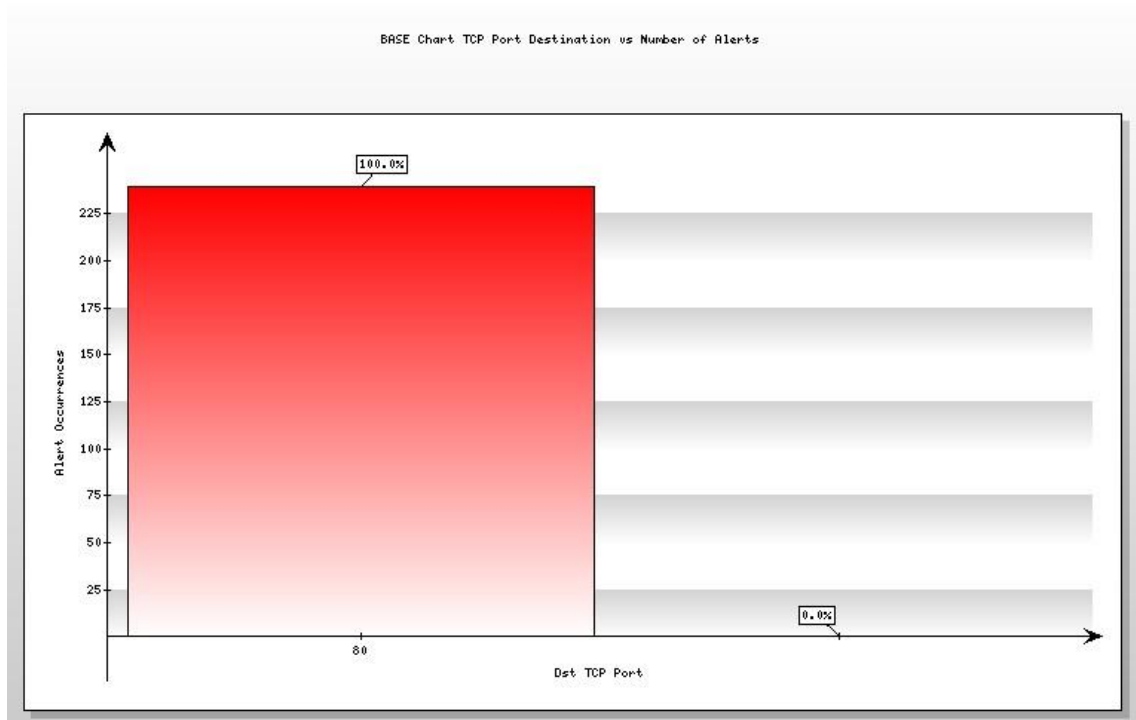


Figura 44: Gráfico alertas ataque DoS en BASE

5. Conclusiones y líneas futuras

Analizando tanto el estudio teórico previo como los resultados obtenidos con Snort, se puede concluir que es un IDS completo y realmente útil para detectar a tiempo las intrusiones. Tiene la gran ventaja de ser un software de código abierto con coste cero de instalación o mantenimiento y, además, está en continua mejora con actualizaciones de reglas o paquetes, en las que han contribuido los usuarios reportando las carencias encontradas desde el año 1998, en el que se lanzó Snort.

Todas sus ventajas lo han convertido en una buena elección para proporcionar seguridad a un sistema, pero también cabe mencionar una serie de aspectos negativos que se han encontrado al trabajar con él, aunque no hayan entorpecido gravemente el curso del proyecto:

Dispone de una variedad de herramientas para representar sus alertas o detecciones de una manera más gráfica que le dan un gran valor. Sin embargo, la mayoría son ajenas a los creadores de Snort y, 22 años tras su lanzamiento, muchas ya han dejado de recibir soporte y actualizaciones, y por ello, no están adecuadas a los nuevos sistemas operativos o presentan incompatibilidades.

Además, necesita una gran personalización previa para establecer a qué máquina o red debe proteger, y ante quienes, ya que no existe una configuración única perfecta para todo tipo de escenarios o empresas. Por ello es necesario que el usuario que vaya a recibir las alertas tenga un alto conocimiento sobre el tema y sepa reaccionar ante una alerta e interpretar los gráficos obtenidos por las herramientas gráficas.

Por otro lado, durante la elaboración del trabajo realizado se han considerado una serie de líneas de investigación interesantes como para tenerlas en cuenta en futuros proyectos:

Una propuesta muy general es descubrir nuevas funciones de Snort, ya que es una herramienta muy versátil que está en continua mejora, por lo que en escenarios diferentes o pasado un tiempo pueden descubrirse otras utilidades interesantes o nuevas alertas integradas, como la descubierta en el ataque del escaneo de puertos realizado.

Se sugiere también encontrar e implementar junto a Snort un software capaz de detectar las alertas automáticamente y actuar, por ejemplo, en el caso del ataque DoS para bloquear la dirección IP atacante al detectar la alerta de

Snort. Es necesario tener en cuenta que se trata estrictamente de un Sistema de Detección de Intrusiones y por ello su trabajo se centra en detectarlas y lanzar alertas para que éstas no pasen por alto, por lo que es necesario complementar a Snort con sistemas de contraataque como este, o realizar de manera manual las acciones pertinentes para detener el ataque que se pueda estar sufriendo, cosa que en una red amplia no sería muy práctico.

Por último, se ha descubierto que en el modo sniffer en tiempo real de Snort puede haber pérdida de paquetes en instantes con mucha carga de tráfico. Por ello otra propuesta de línea de investigación podría consistir en el estudio de cómo afectan las condiciones del tráfico al coste computacional, y la relación con la pérdida de capacidad de detección. También podría estudiarse el trabajo con firewalls diseñados para restarle trabajo a Snort y que realicen un filtrado en primer plano para optimizar recursos computacionales una vez más.

Bibliografía

- [1] RedIRIS. *Sistemas de detección de intrusos*. 2002. Disponible en: <https://www.rediris.es/cert/doc/unixsec/node26.html>
- [2] Martí JUNCOSA. *Fragmentación IP*. 2019. Disponible en: <https://aprendederedes.com/redes/ip/fragmentacion-ip/>
- [3] TOAD. *Introducción a TCP/IP*. 2005. Disponible en: https://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Introduccion_a_TCPIP.pdf
- [4] RedIRIS. *Ataques remotos*. 2002. Disponible en: <https://www.rediris.es/cert/doc/unixsec/node25.html>
- [5] Alfon. *Sistemas de Detección de intrusos y Snort (I)*. 2007. Disponible en: <https://seguridadyredes.wordpress.com/2007/12/28/sistemas-de-deteccion-de-intrusos-y-snort-i/>
- [6] Alejandro SÁNCHEZ. *Mejores IDS Opensource para Detección de Intrusiones*. 2018. Disponible en: <https://protegermipc.net/2017/02/22/mejores-ids-opensource-deteccion-de-intrusiones/>
- [7] Snort. *Snort*. 2013. Disponible en: <https://snortudonar.wordpress.com/snort-2/>
- [8] Suricata. *Suricata*. Disponible en: <https://suricata-ids.org/>
- [9] Rubén VELASCO. *Suricata 3.0, novedades de este nuevo monitor de seguridad libre*. 2016. Disponible en: <https://www.redeszone.net/2016/01/28/suricata-3-0-novedades-de-este-nuevo-monitor-de-seguridad-libre/>
- [10] Aaron TF. *Snort vs Suricata Feature Comparison*. 2019. Disponible en: <https://tacticalflex.zendesk.com/hc/en-us/articles/360010678893-Snort-vs-Suricata>
- [11] Alisa ESAGE. *Herramientas para afinar y complementar Snort*. 2015. Disponible en: <https://noticiasseguridad.com/importantes/herramientas-para-afinar-y-complementar-ids-snort/>
- [12] Jane RUOSTEMAA. *How to install Snort on Ubuntu*. 2019. Disponible en: <https://upcloud.com/community/tutorials/install-snort-ubuntu/>

- [13] Alfon. *IDS Policy Manager. Configuración y Funcionamiento. Administración múltiples sensores y políticas snort*. 2008. Disponible en: <https://seguridadyredes.wordpress.com/2008/02/26/ids-policy-manager-configuracion-y-funcionamiento-administracion-multiples-sensores-y-politicas-snort/>
- [14] Cybermaggedon. *Cyberprobe 2.5.1*. 2020. Disponible en: <https://cybermaggedon.github.io/cyberprobe-docs/cyberprobe.html>
- [15] Kellep CHARLES. *Using Metasploit to Conduct NMAP Scans*. 2018. Disponible en: <https://www.securityorb.com/featured/using-metasploit-to-conduct-nmap-scans/>
- [16] Pentestit. *Brute-force attacks with Kali Linux*. 2019. Disponible en: https://medium.com/@Pentestit_ru/brute-force-attacks-using-kali-linux-49e57bb89259
- [17] Rubén VELASCO. *Hping3: Manual de utilización de esta herramienta para manipular paquetes TCP/IP*. 2015. Disponible en: <https://www.redeszone.net/gnu-linux/hping3-manual-de-utilizacion-de-esta-herramienta-para-manipular-paquetes-tcp-ip/>
- [18] David CANTÓN. *DoS: Capa de Infraestructura*. 2015. Disponible en: <https://www.incibe-cert.es/blog/dos-capa-infraestructura/>

